APOSTILA KIT

AUTOMAÇÃO RESIDENCIAL

COM MÓDULO WI-FI NODEMCU

WWW.ELETROGATE.COM

Apostila Eletrogate

KIT Automação Residencial

Esta apostila acompanha o Kit **Automação Residencial** da Eletrogate, e contém conteúdos relacionados a todos os componentes dos kits.

Sumário

Parte I - Revisão de circuitos elétricos e componentes básicos

Introdução

A primeira parte da apostila faz uma revisão sobre circuitos elétricos, com destaque para questões práticas de montagem e segurança que surgem no dia-a-dia do usuário do Kit Automação Residencial.

O conteúdo de circuitos elétricos aborda divisores de tensão e corrente, conversão de níveis lógicos, grandezas analógicas e digitais, níveis de tensão e cuidados práticos de montagem.
Em relação aos componentes básicos, é feita uma breve discussão sobre resistores elétricos, capacitores, leds, diodos, chaves e protoboards.

O NodeMCU ESP8266 é o principal componente do Kit e é discutido e introduzido em uma seção à parte, na Parte II da apostila.

Todos os conteúdos da Parte I são focados na apostila Automação Residencial, tendo em vista a utilização adequada dos componentes e da realização prática de montagens pelos usuários. No entanto, recomenda-se a leitura das referências indicadas ao final de cada seção para maior aprofundamento.

O leitor veterano, já acostumado e conhecedor dos conceitos essenciais de eletrônica e eletricidade, pode pular a Parte I e ir direto a Parte II, na qual são apresentadas uma seção de exemplo de montagem para cada sensor ou componente importante da apostila.

Algum conhecimento prévio é bem-vindo, mas não é necessário para utilização do kit. Caso seja seu primeiro contato, nós recomendamos que antes de fazer as montagens você leia esses três artigos do *http://blog.eletrogate.com/*.

- *http://blog.eletrogate.com/arduino-primeiros-passos/*
- *http://blog.eletrogate.com/programacao-arduino-parte-1/*
- *http://blog.eletrogate.com/programacao-arduino-parte-2/*

Preparado? Vamos começar!

Revisão de circuitos elétricos

A apostila de Automação Residencial, bem como todas as outras apostilas que tratam de Arduino e eletrônica em geral, tem como conhecimento de base as teorias de circuitos elétricos e de eletrônica analógica e digital.

Do ponto de vista da teoria de circuitos elétricos, é importante conhecer os conceitos de grandezas elétricas: Tensão, corrente, carga, energia, potência elétrica. Em todos os textos sobre Arduino ou qualquer assunto que envolva eletrônica, você sempre terá que lidar com esses termos. Para o leitor que se inicia nessa seara, recomendamos desde já que mesmo que a eletrônica não seja sua área de formação, que conheça esses conceitos básicos.

Vamos começar pela definição de "circuito elétrico". ***Um circuito elétrico/eletrônico é uma interconexão de elementos elétricos/eletrônicos***. Essa interconexão pode ser feita para atender a uma determinada tarefa, como acender uma lâmpada, acionar um motor, dissipar calor em uma resistência e tantas outras.

O circuito pode estar **energizado** ou **desenergizado**. Quando está energizado, é quando uma fonte de tensão externa ou interna está ligada aos componentes do circuito. Nesse caso, uma corrente elétrica fluirá entre os condutores do circuito. Quando está desenergizado, a fonte de tensão não está conectada e não há corrente elétrica fluindo entre os condutores.

Mas atenção, alguns elementos básicos de circuitos, como os capacitores ou massas metálicas, são elementos que armazenam energia elétrica. Em alguns casos, mesmo não havendo fonte de tensão conectada a um circuito, pode ser que um elemento que tenha energia armazenada descarregue essa energia dando origem a uma corrente elétrica transitória no circuito. Evitar que elementos do circuito fiquem energizados mesmo sem uma fonte de tensão, o que pode provocar descargas elétricas posteriores (e em alguns casos, danificar o circuito ou causar choques elétricos) é um dos motivos dos sistemas de aterramento em equipamentos como osciloscópios e em instalações residenciais, por exemplo.

Em todo circuito você vai ouvir falar das grandezas elétricas principais, assim, vamos aprender o que é cada uma delas.

Carga e corrente elétrica

A grandeza mais básica nos circuitos elétricos é a carga elétrica. Carga é a propriedade elétrica das partículas atômicas que compõem a matéria, e é medida em Coulombs. Sabemos da física elementar que a matéria é constituída de elétrons, prótons e nêutrons. **A carga elementar é a carga de 1 elétron, que é igual a 1,602 x10^{-19} C**.

Do conceito de carga elétrica advém o **conceito de corrente elétrica, que nada mais é do que a taxa de variação da carga em relação ao tempo**, ou seja, quando você tem um fluxo de carga em um condutor, a quantidade de carga (Coulomb) que atravessa esse condutor por unidade de tempo, é chamada de corrente elétrica. A medida utilizada para corrente é o **Ampére(A).**

Aqui temos que fazer uma distinção importante. Existem corrente elétrica contínua e alternada:

- **Corrente elétrica contínua:** É uma corrente que permanece constante e em uma única direção durante todo o tempo.
- **Corrente elétrica alternada**: É uma corrente que varia senoidalmente (ou de outra forma) com o tempo.

Com o NodeMCU ESP8266, lidamos com correntes elétricas contínuas, pois elas fluem sempre em uma mesma direção. É diferente da corrente e tensão elétrica da tomada de sua casa, que são alternadas.

Ou seja, os seus circuitos com NodeMCU ESP8266 sempre serão alimentados com grandezas contínuas (corrente e tensão contínuas).

Tensão elétrica

Para que haja corrente elétrica em um condutor, é preciso que os elétrons se movimentam por ele em uma determinada direção, ou seja, é necessário "alguém" para transferir energia para as cargas elétricas para movê-las. Isso é feito por uma força chamada **força eletromotriz (FEM),** tipicamente representada por uma bateria. Outros dois nomes comuns para força eletromotriz são **tensão elétrica** e **diferença de potencial.**

O mais comum é você ver apenas “*tensão*” nos artigos e exemplos com Arduino. Assim, definindo formalmente o conceito: Tensão elétrica é a energia necessária para mover uma unidade de carga através de um elemento, e é medida em **Volts (V).**

Potência e energia

A tensão e a corrente elétrica são duas grandezas básicas, e juntamente com a potência e energia, são as grandezas que descrevem qualquer circuito elétrico ou eletrônico. A potência é definida como a variação de energia (que pode estar sendo liberada ou consumida) em função do tempo, e é medida em **Watts (W).** A potência está associada ao calor que um componente está dissipando e a energia que ele consome.

Nós sabemos da vida prática que uma lâmpada de 100W consome mais energia do que uma de 60W. Ou seja, se ambas estiverem ligadas por 1 hora por exemplo, a lâmpada de 100W vai implicar numa conta de energia mais cara.
A potência se relaciona com a tensão e corrente pela seguinte equação:

$$P = V \times I$$

Essa é a chamada potência instantânea. Com essa equação você saber qual a potência dissipada em um resistor por exemplo, bastando que saiba a tensão nos terminais do resistor e a corrente que flui por ele. O conceito de potência é importante pois muitos hobistas acabam não tendo noção de quais valores de resistência usar, ou mesmo saber especificar componentes de forma adequada.

Um resistor de 33 Ω de potência igual a 1/4W, por exemplo, não pode ser ligado diretamente em circuito de 5V, pois nesse caso a potência dissipada nele seria maior que a que ele pode suportar.

Vamos voltar a esse assunto em breve, por ora, tenha em mente que é importante ter uma noção da potência dissipada ou consumida pelos elementos de um circuito que você vá montar.

Por fim, a energia elétrica é o somatório da potência elétrica durante todo o tempo em que o circuito esteve em funcionamento. A energia é dada em **Joules (J)** ou **Wh (watt-hora).** A unidade Wh é interessante pois mostra que a energia é calculada multiplicando-se a potência pelo tempo (apenas para os casos em que a potência é constante).

Essa primeira parte é um pouco conceitual, mas é importante saber de onde veio toda a terminologia que você sempre vai ler nos manuais e artigos na internet. Na próxima seção, vamos discutir os componentes básicos de circuito que compõem o Kit Automação Residencial.

Conversão de níveis lógicos e alimentação correta de circuitos

É muito comum que hobbistas e projetistas em geral acabam por cometer alguns erros de vez em quando. Na verdade, mesmo alguns artigos na internet e montagens amplamente usadas muitas vezes acabam por não utilizar as melhores práticas de forma rigorosa. Isso acontece muito no caso dos níveis lógicos dos sinais usados para interfacear o Arduino com outros circuitos.

Como veremos na seção de apresentação do NodeMCU ESP8266, o mesmo é alimentado por um cabo USB ou uma fonte externa de 5V. Os sinais lógicos das portas de saída(I/Os) do NodeMCU, variam entre 0 e 5V.

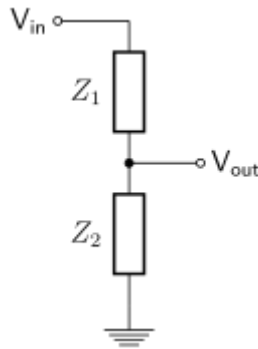
Isso significa que quando você quiser usar o seu NodeMCU ESP8266 com um sensor ou CI que trabalhe com 3.3V, é preciso fazer a adequação dos níveis de tensão, pois se você enviar um sinal de 5V (saída do NodeMCU) em um circuito de 3.3V (CI ou sensor), você poderá queimar o pino daquele componente.
Em geral, sempre que dois circuitos que trabalham com níveis de tensão diferentes forem conectados, é preciso fazer a conversão dos níveis lógicos. O mais comum é ter que abaixar as saídas de 5V para 3.3V. Subir os sinais de 3.3V para 5V na maioria das vezes não é necessário pois o NodeMCU entende 3.3V como nível lógico alto, isto é, equivalente a 5V.

Para fazer a conversão de níveis lógicos você tem duas opções:

- Usar um divisor de tensão;
- Usar um *CI conversor de níveis lógicos*;

O divisor de tensão é a solução mais simples, mas usar um CI conversor é mais elegante e é o ideal. O divisor de tensão consiste em dois resistores ligados em série (Z1 e Z2) , em que o sinal de 5V é aplicado a o terminal de um deles. O terminal do segundo resistor é ligado ao GND, e o ponto de conexão entre os dois resistores é a saída do divisor, cuja tensão é dada pela seguinte relação:

$$V_{\text{out}} = \frac{Z_2}{Z_1 + Z_2} \cdot V_{\text{in}}$$

Em que Z1 e Z2 são os valores dos resistores da figura abaixo.

Um divisor de tensão muito comum é fazer Z1 igual 330 Ω e Z2 igual 680 Ω. Dessa forma a saída Vout fica sendo 3.336 V. Como no Kit Automação Residencial não há resistor de 680 Ω, você pode ligar um de 330 Ω como Z1 e dois de 330 Ω como Z2. Os dois de 330 ligados em série formam um resistor de 660 ohm, o que resulta numa saída de 3.33V.

Vamos exemplificar como fazer um divisor de tensão como esse na seção de exemplos da parte II da apostila.

Revisão de componentes eletrônicos

O Kit Automação Residencial possui os seguintes componentes básicos para montagens de circuitos:

- LEDs Vermelho/ Verde/ Amarelo,
- Resistores 330Ω/ 1KΩ/ 10KΩ,
- Diodos 1N4007,
- Potenciômetro 10KΩ,
- Capacitores Cerâmicos 10nF/ 100nF,
- Capacitores Eletrolíticos 10uF/ 100uF,
- Chave Táctil (Push-Button).

Vamos revisar a função de cada um deles dentro de um circuito eletrônico e apresentar mais algumas equações fundamentais para ter em mente ao fazer suas montagens.

Resistores elétricos

Os resistores são componentes que se opõem à passagem de corrente elétrica, ou seja, oferecem uma resistência elétrica. Dessa forma, quanto maior for o valor de um resistor, menor será a corrente elétrica que fluirá por ele e pelo condutor a ele conectada. A unidade de resistência elétrica é o **Ohm(Ω)**, que é a unidade usada para especificar o valor dos resistores.

Os resistores mais comuns do mercado são construídos com fio de carbono e são vendidos em várias especificações. Os resistores do Kit são os tradicionais de 1/4W e 10% de tolerância. Isso significa que eles podem dissipar no máximo 1/4W (0,25 watts) e seu valor de resistência pode variar em até 10%. O resistor de 1KΩ pode então ter um valor mínimo de 900Ω e um valor máximo de 1100Ω.

Em algumas aplicações você pode precisar de resistores com precisão maior, como 5% ou 1%. Em outros casos, pode precisar de resistores com potência maior, como 1/2W ou menor, como 1/8W. Essas variações dependem da natureza de cada circuito.

Em geral, para as aplicações típicas e montagens de prototipagem que podem ser feitas com o Kit Automação Residencial, os resistores tradicionais de 1/4W e 10% de tolerância são suficientes.

Outro ponto importante de se mencionar aqui é a Lei de Ohm, que relaciona as grandezas de tensão, corrente e resistência elétrica. A lei é dada por:

$$V = R \times I$$

Ou seja, se você sabe o valor de um resistor e a tensão aplicada nos seus terminais, você pode calcular a corrente elétrica que fluirá por ele. Juntamente com a equação para calcular potência elétrica, a lei de Ohm é importante para saber se os valores de corrente e potência que os resistores de seu circuito estão operando estão adequados.

Para fechar, você deve estar se perguntando, como saber o valor de resistência de um resistor? Você tem duas alternativas: Medir a resistência usando um multímetro ou determinar o valor por meio do código de cores do resistor.

Se você pegar um dos resistores do seu kit, verá que ele possui algumas faixas coloridas em seu corpo. Essas faixas são o código de cores do resistor. As duas primeiras faixas dizem os dois primeiros algarismos decimais. A terceira faixa colorida indica o multiplicador que devemos usar. E a última faixa, que fica um pouco mais afastada, indica a tolerância.

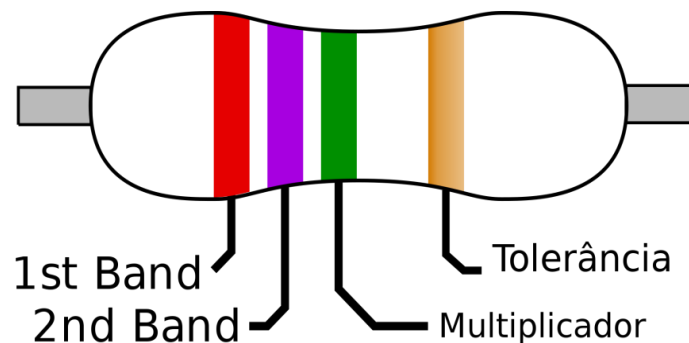



Figura 1: Faixas coloridas em um resistor

Na figura 2, apresentamos o código de cores para resistores. Cada cor está associada a um algarismo, um multiplicador e uma tolerância, conforme a tabela. Com a tabela você pode determinar a resistência de um resistor sem ter que usar o multímetro.

Mas atenção, fazer medições com o multímetro é recomendado, principalmente se o componente já tiver sido utilizado, pois o mesmo pode ter sofrido algum dano ou mudança que não esteja visível.

Cor	Dígito	Multiplicador	Tolerância
Prata	-	x 0,01	± 10%
Dourado	-	x 0,1	± 5%
Preto	0	x 1	-
Marrom	1	x 10	± 1%
Vermelho	2	x 100	± 2%
Laranja	3	x 1K	-
Amarelo	4	x 10K	-
Verde	5	x 100K	± 0,5%
Azul	6	x 1M	± 0,25%
Violeta	7	x 10M	± 0,1%
Cinza	8	-	± 0,05%
Branco	9	-	-

Figura 2: Código de cores para resistores

Aplicando a tabela da figura 2 na imagem da figura 1, descobrimos que o resistor é de 2,7MΩ (Mega Ω) com tolerância de 5% (relativo à cor dourada da última tira).

Capacitores

Os capacitores são os elementos mais comuns nos circuitos eletrônicos depois dos resistores. São elementos que armazenam energia na forma de campos elétricos. Um capacitor é constituído de dois terminais condutores e um elemento dielétrico entre esses dois terminais, de forma que quando submetido a uma diferença de potencial, um campo elétrico surge entre esses terminais, causando o acúmulo de cargas positivas no terminal negativo e cargas negativas no terminal positivo.

São usados para implementar filtros, estabilizar sinais de tensão, na construção de fontes retificadores e várias outras aplicações.

O importante que você deve saber para utilizar o Kit é que os capacitores podem ser de quatro tipos:

- Eletrolíticos,
- Cerâmicos,
- Poliéster,
- Tântalo.

Capacitor eletrolítico

As diferenças de cada tipo de capacitor são a tecnologia construtiva e o material dielétrico utilizado. Capacitores eletrolíticos são feitos de duas longas camadas de alumínio (terminais) separadas por uma camada de óxido de alumínio (dielétrico). Devido a sua construção, eles possuem **polaridade,** o que significa que você obrigatoriamente deve ligar o terminal positivo (quem tem uma "perninha" maior) no pólo positivo da tensão de alimentação, e o terminal negativo (discriminado no capacitor por uma faixa com símbolos de "-") obrigatoriamente no pólo negativo da bateria. Do contrário, o capacitor será danificado.

Capacitores eletrolíticos costumam ser da ordem de microFarad, sendo o **Farad** a unidade de medida de capacitância, usada para diferenciar um capacitor do outro. A Figura 3 ilustra um típico capacitor eletrolítico.

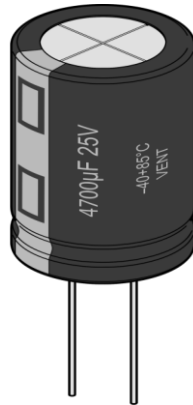



Figura 3: Capacitor eletrolítico 4700 microFarads / 25 V

Capacitores cerâmicos

Capacitores cerâmicos não possuem polaridade, e são caracterizados por seu tamanho reduzido e por sua cor característica, um marrom claro um tanto fosco. Possuem capacitância da ordem de **picoFarad**. Veja nas imagens abaixo um típico capacitor cerâmico e sua identificação:

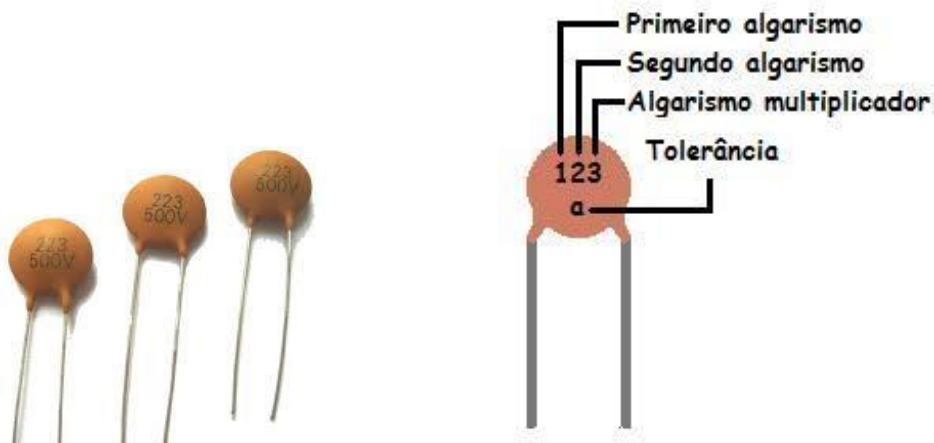



Figura 4: Capacitores cerâmicos

Na imagem da esquerda, os capacitores possuem o valor de **22 nanoFarads** para a faixa de tensão de até 500V.

223 = 22 x 1000 = 22.000 pF = 22 nF

Por fim, há também os capacitores de poliéster e de tântalo. No Kit Automação Residencial, você receberá apenas exemplares de capacitores eletrolíticos e cerâmicos.

Diodos e Leds

Diodos e Leds são tratados ao mesmo tempo pois tratam na verdade do mesmo componente. Diodos são elementos semicondutores que só permitem a passagem de corrente elétrica em uma direção.

São constituídos de dois terminais, o **Ânodo(+)** e o **Cátodo(-),** sendo que para que possa conduzir corrente elétrica, é preciso conectar o Ânodo na parte positiva do circuito, e o Cátodo na parte negativa. Do contrário, o diodo se comporta como um circuito aberto.

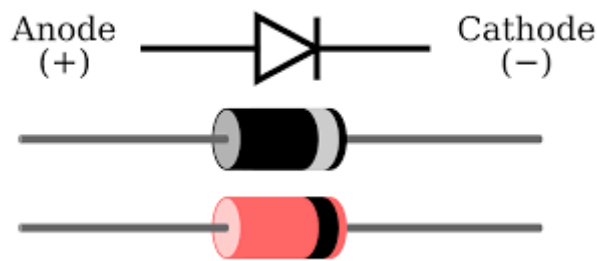



Figura 4: Diodo e seus terminais

Na figura 4, você pode ver que o componente possui uma faixa indicadora do terminal de catodo. O diodo do kit Automação Residencial é um modelo tradicional e que está no mercado há muitos anos, o 1N4007.

O LED é um tipo específico de diodo - **Light Emitter Diode**, ou seja, diodo emissor de luz. Trata-se de um diodo que quando polarizado corretamente, emite luz para o ambiente externo. O Kit Automação Residencial vem acompanhado de leds na cor vermelha, verde e amarelo, as mais tradicionais.

Nesse ponto, é importante você saber que sempre deve ligar um led junto de um resistor, para que a corrente elétrica que flua pelo led não seja excessiva e acabe por queimá-lo. Além disso, lembre-se que por ser um diodo, o led só funciona se o Ânodo estiver conectado ao pólo positivo do sinal de tensão.

Para identificar o Ânodo do Led, basta identificar a perna mais longa do componente, como na imagem abaixo:

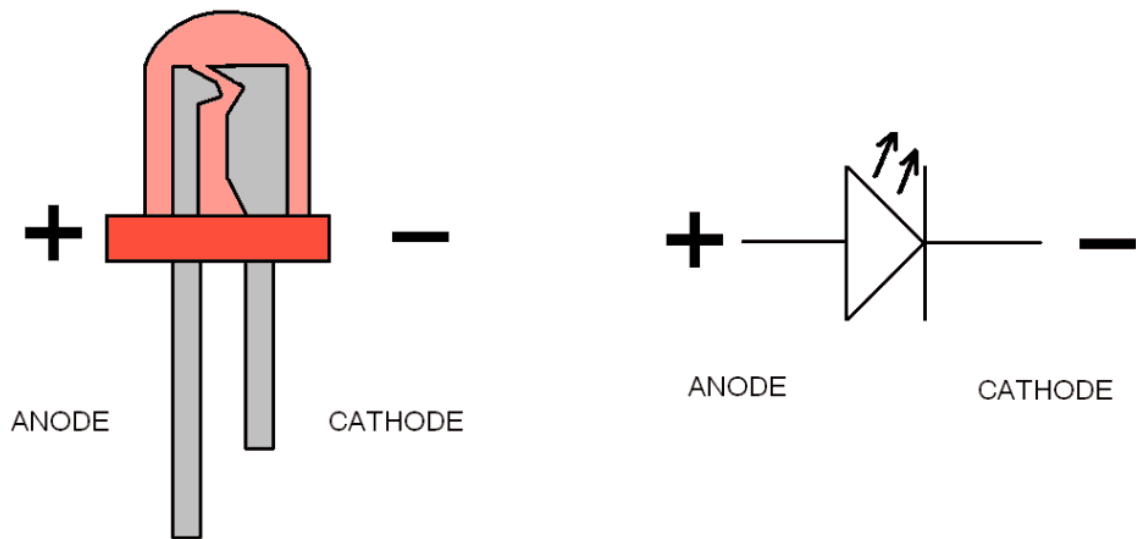



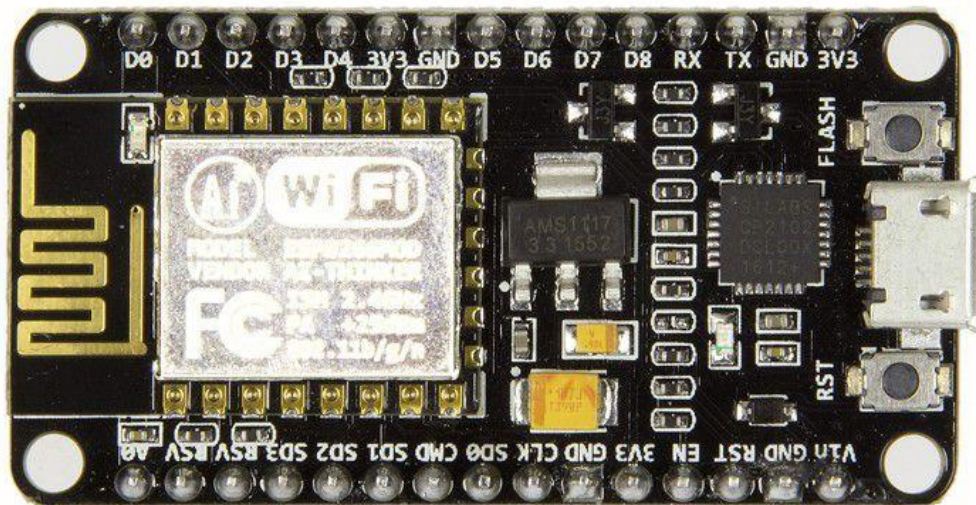
Figura 5: Terminais de um Led. Créditos: *Build-eletronic-circuits.com*

Os demais componentes da apostila, potenciômetro e push-buttons, serão explicados em seções de exemplos, nas quais iremos apresentar um circuito prático para montagem onde será explicado o funcionamento dos mesmos.

Os sensores do Kit Automação Residencial, quais sejam: Sensor de Luz LDR e Sensor de Temperatura NTC, também terão uma seção de exemplo dedicada a cada um deles.

Parte II – NodeMCU ESP8266

O que é o Módulo NodeMCU-ESP12 ?

O Módulo NodeMCU é uma placa que foi criada para facilitar o desenvolvimento de aplicações para o módulo ESP8266 ESP-12.

Como é possível ver na foto, existe um módulo ESP-12 soldado na placa. Nessa placa, já existem todos os circuitos necessários para fazer o ESP-12 funcionar – interface Serial-USB, regulador de tensão, leds indicadores, botões de controle (Reset e Flash) e barramentos de pinos para permitir o uso em Protoboards.

A grande vantagem dessa placa NodeMCU é que ela funciona como se fosse um Arduino. Você conecta o cabo USB, e com a IDE Arduino você carrega seus programas na placa. Nem precisa pressionar os botões!

Esse são os modelos de NodeMCU-ESP12 mais comuns, atualmente:

- NodeMCU ESP12-N,
- NodeMCU ESP12-E.

Para a alimentação do ESP-12, existe um regulador de 3,3V AMS1117 (corrente máxima de 1A).

Considerando que o consumo máximo de um ESP-12 é de aproximadamente 200 mA, sobra uma corrente disponível de 800 mA. Mas se for usar a tensão de 3,3V do próprio regulador para alimentar um outro dispositivo externo, é recomendado que nunca ultrapasse os 500 mA.

A alimentação dessa placa pode ser feita através do próprio conector USB (5,0V) ou então através do pino VIN (EXT) , com uma alimentação regulada de 5,0V .

Apesar do Regulador AMS1117 aceitar tensões de até 9V na entrada, é recomendado que, se for alimentar a placa através deste pino, use sempre uma fonte regulada de 5V, pois assim nunca sobre-aquecerá o regulador! Evitando um possível defeito no mesmo.

Não é recomendado que se use o pino 3,3V REG para a entrada de alimentação da placa. Esse pino é a saída do regulador e não a entrada. Porém o site da ESPRESSIF especifica que a alimentação pode ser também através do pino 3,3V REG.

Uma observação importante: as opções de alimentação são mutuamente exclusivas, isto é, somente poderá usar uma opção (USB, 5V ou 3,3V) . Não use mais de uma opção, pois poderá danificar algum componente da placa. Não se esqueça de conectar o GND da fonte no GND da placa.

O Chip da interface Serial-USB é o CP2102 da Silicon Labs. Ele suporta USB 2.0. (não suporta USB 3.0). Baud Rates até 1 MBps. Na placa NodeMCU, a alimentação 3,3V do CP2102 é feita através do regulador AMS1117. O consumo máximo de corrente desse chip é de apenas 26 mA. O CP2102 possui um regulador interno de 3,3V, mas nessa placa NodeMCU, este regulador não é usado.

Para fazer a comunicação serial-USB com o seu computador é necessário que, antes de conectar o cabo USB, instale os drivers do CP2102.

Para fazer o download desses drivers para Windows, Macintosh OSX, Linux e Android, use o link abaixo:

https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers

A pinagem do módulo NodeMCU ESP-12

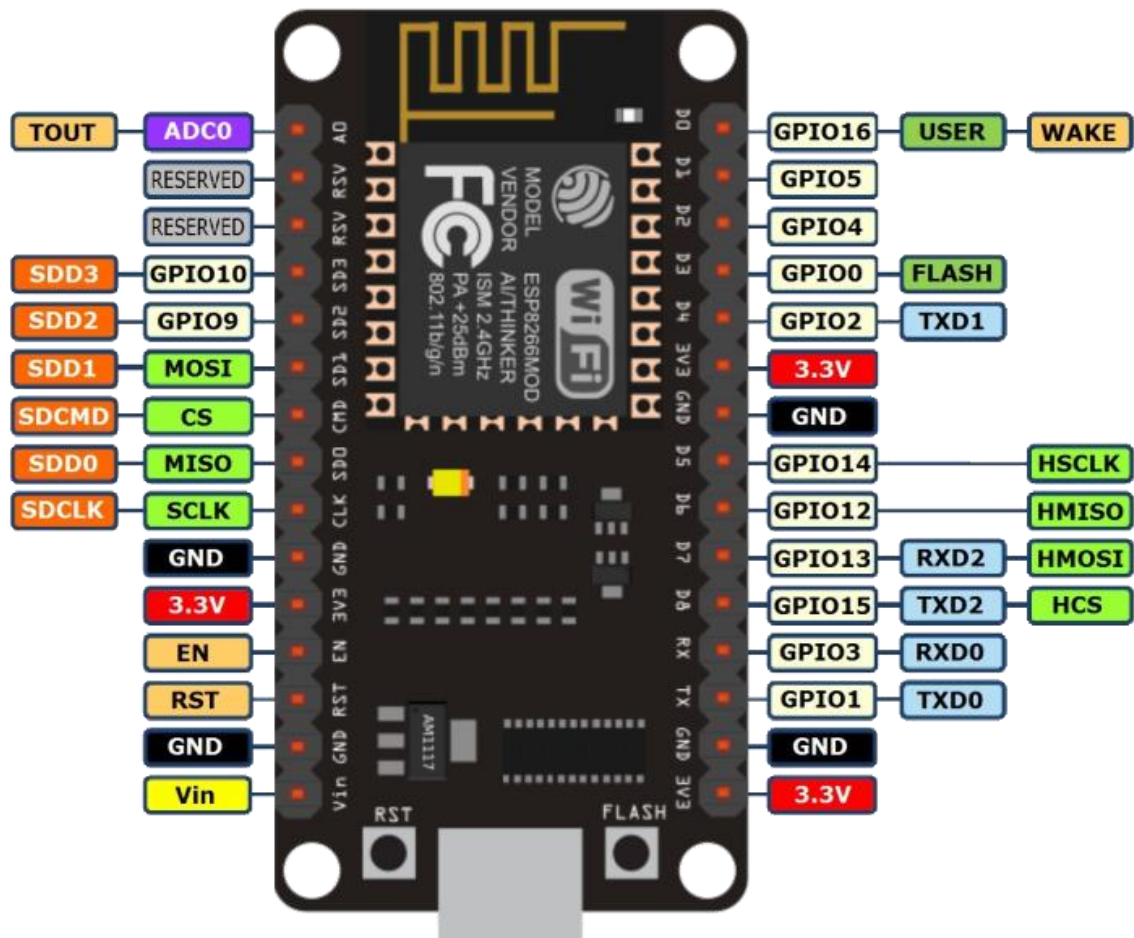



Figura 6: Pinagem do Módulo NodeMCU ESP-12 | Fonte: NodeMCU ESP-12 (www.arduining.com)

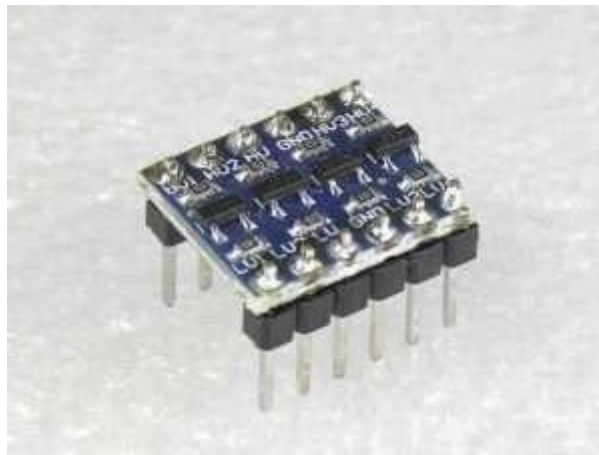
- **Led indicador Azul** – está conectado no pino GPIO16. Um pulso LOW(0V) acionará o led.
- **Led Indicador ESP-12** – pisca quando a memória Flash está sendo gravada
- **Botão de RST** – dá um pulso LOW (0V) no pino -RST (reset) reboot no módulo ESP-12.
- **Botão de FLASH** – dá um pulso LOW (0V) no pino GPIO0 – permite a gravação do programa no ESP-12.

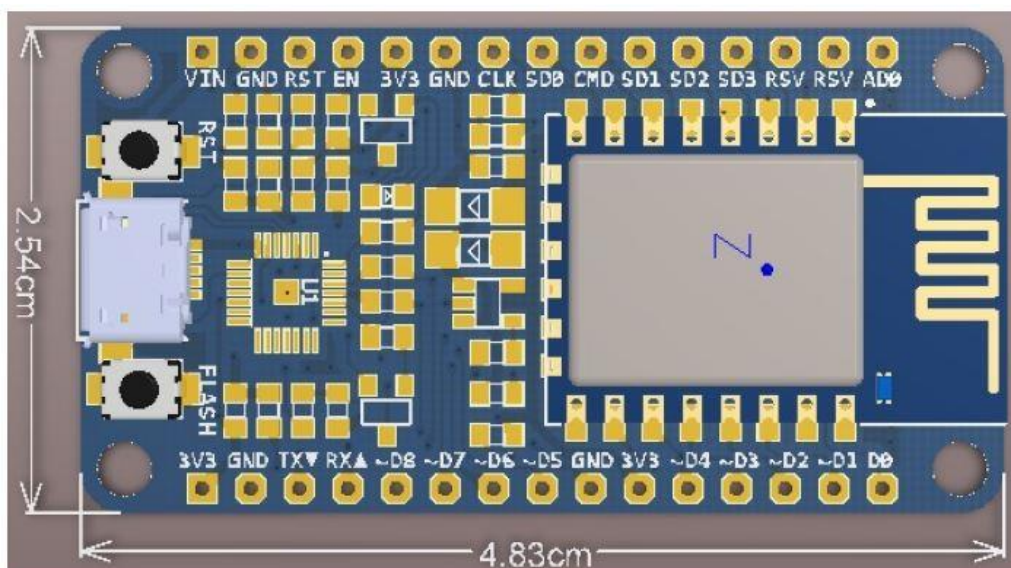
Observação interessante : Os pinos DTR e RTS do Chip CP2102, controlam o pino ENA (reset chip ESP-12) e o pino GPI0 (Load program). Portanto, ao gravar um programa com a IDE Arduino, não é necessário pressionar o botão de BOOT (Load).

<u>Aviso importante</u>: A pinagem na placa NodeMCU é diferente da pinagem interna, utilizada na programação! Quando for programar, declare os pinos com seus números internos, por exemplo, se for utilizar o pino D1, declare ele como 5.

Função dos Pinos

Todos os pinos GPIOs podem ser entradas ou saídas dependendo da configuração dos mesmos. Não ultrapasse a corrente de 12 mA em cada porta dessas, pois poderá danificar o chip.

O recomendado é 6 mA. O nível de tensão a ser usado nessas portas não deverá ultrapassar os 3,3V. Se for necessário conectar o NodeMCU a outro dispositivo de tensão maior, como um Arduino ou módulos de 5V, use conversores bidirecionais de tensão como o do link abaixo ou use divisores resistivos de tensão.

Conversor Bidirecional de tensão

Essas são as dimensões do Módulo NodeMCU ESP-12:

Fonte: User Manual for ESP12 Kit

O NodeMCU ESP12 tem duas fileiras de 15 pinos (total 30 pinos). A distância entre as duas fileiras é grande (2,30 cm), mas poderá inseri-lo em um Protoboard padrão. Os pinos RESERVED não deverão ser usados, como o nome já diz, são reservados.

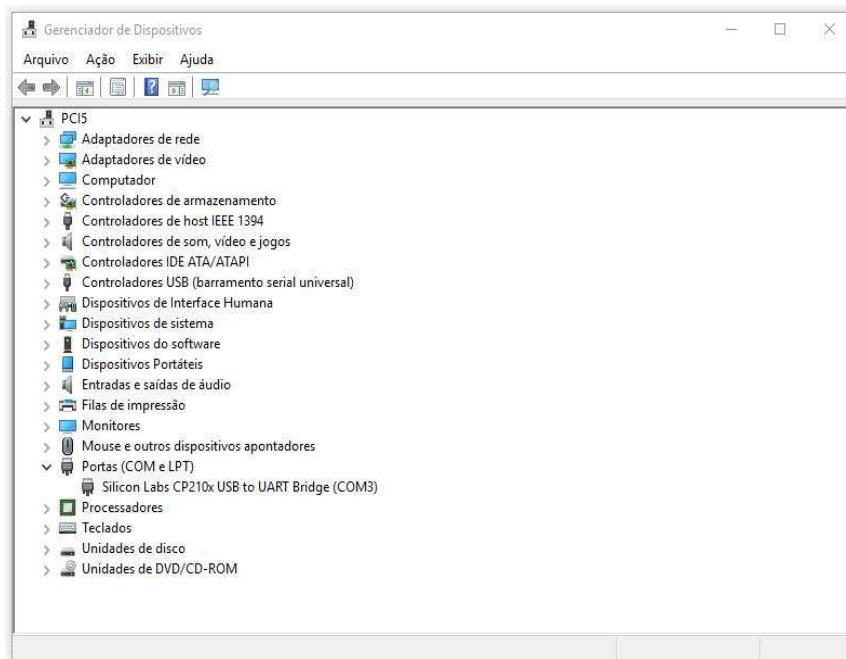
- **VIN** – Este é o pino de alimentação externa (recomendado é 5,0V/1A). Pode-se utilizar até 9V, mas o regulador da placa deverá esquentar. Não use-o se estiver usando USB.
- **GND** – Este é o terra da placa. Não se esqueça de conectá-lo ao terra de outros dispositivos.
- **RST** – Reset do módulo ESP-12. Nível LOW (0V) dá um reboot na placa.
- **EN** – Enable ativa o módulo ESP-12 quando o nível for HIGH (3,3V).
- **3.3V** – Saída do regulador interno de 3,3V para alimentar outro dispositivo. Não use mais do que 500 mA de corrente.
- **CLK** – interface SPI (clock) – pino SCLK (GPIO6).
- **SD0** – interface SPI (master in serial out) – pino MISO (GPIO7).
- **CMD** – interface SPI (chip select) – pino CS (GPIO11).
- **SD1** – interface SPI (master out serial in) – pino MOSI (GPIO8).
- **SD2** – pino GPIO9 – pode ser usado também para comunicação com SD Card (SDD2).
- **SD3** – pino GIPO10 – pode ser usado também para comunicação com SD Card (SDD3).
- **RSV** – reservado (não use).
- **ADC0** – pino de entrada do conversor analógico digital ADC de 10 bits. Tensão máxima de 1,1V (variação do valor digital – 0 a 1024).
- **D0** – pino GIPO16 – pode ser usado para acordar (WAKE UP) o ESP8266 em modo sono profundo (Deep sleep mode).
- **D1** – pino GPIO5 – entrada ou saída.
- **D2** – pino GIPO4 – entrada ou saída.
- **D3** – pino GPIO0 – é usado também para controlar o upload do programa na memória Flash. Está conectado no botão FLASH.
- **D4** – pino GPIO2 – UART_TXD1 quando carregando o programa na memória FLASH.
- **D5** – pino GPIO14 – pode ser usado em SPI de alta velocidade (HSPI-SCLK).
- **D6** – pino GPIO_12 – pode ser usado em SPI de alta velocidade (HSPI-MISO).
- **D7** – pino GPIO13 – pode ser usado em SPI de alta velocidade (HSPI-MOSI) ou UART0_CTS.
- **D8** – pino GPIO15 – pode ser usado em SPI de alta velocidade (HSPI-CS) ou UART0_RTS.
- **RX** – pino GPIO3 – U0RXD quando carregando o programa na memória FLASH.
- **TX** – pino GIPO1 – U0TXD quando carregando o programa na memória FLASH.

Conectando o Módulo no seu computador

A Placa NodeMCU ESP12 deve ser conectada no seu computador, usando um cabo USB com conector micro-USB. Qualquer porta USB poderá ser usada, mas dê a preferência para uma porta USB que suporte a corrente de 500mA ou mais.

Antes de conectar o cabo, como eu já informei, instale os drivers do chip Serial-USB. Aguarde o reconhecimento da placa pelo Windows.

Para descobrir qual porta COM será usada pela Plataforma de Desenvolvimento (IDE) do ESP8266, acesse o Gerenciador de dispositivos e identifique a COM configurada.

Digite Gerenciador de dispositivos na caixa de pesquisa do Windows e selecione-o. No meu PC, a porta configurada foi a COM3.

O NodeMCU é configurado no gerenciador de dispositivos

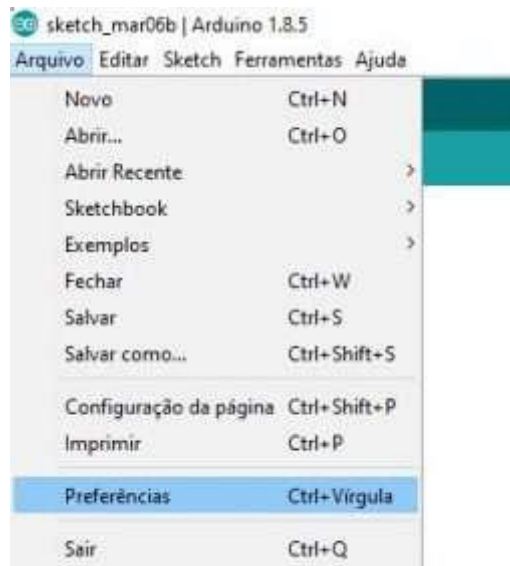
Procedimentos para instalar NodeMCU na Arduino IDE

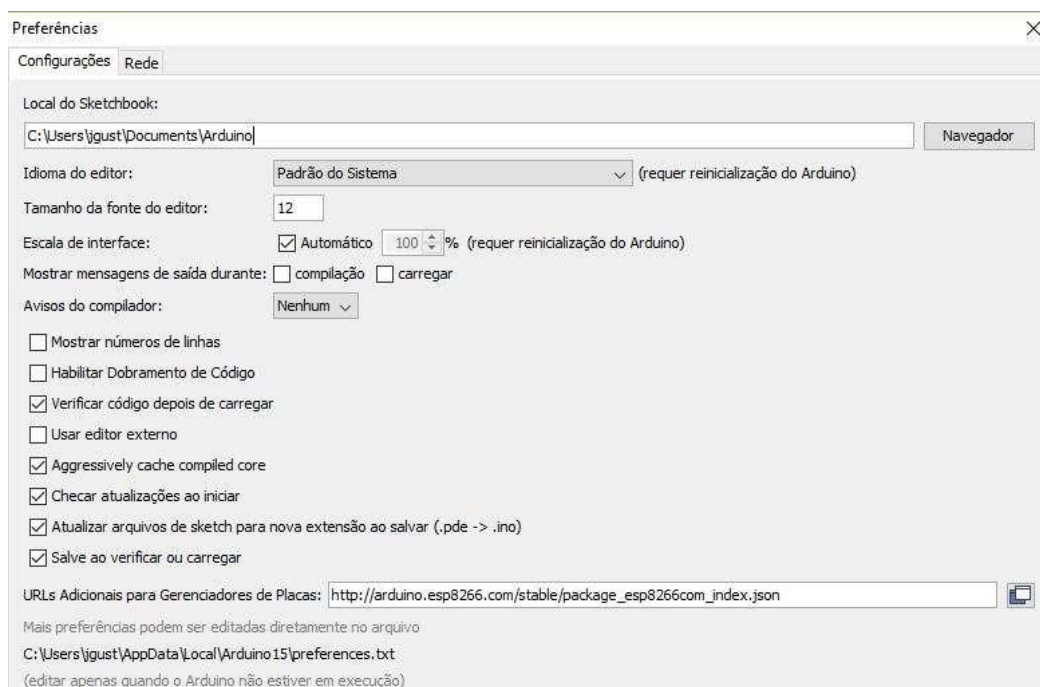
Existem alguns procedimentos para a instalação do NodeMCU na Arduino IDE. Aqui será utilizado o mais simples.

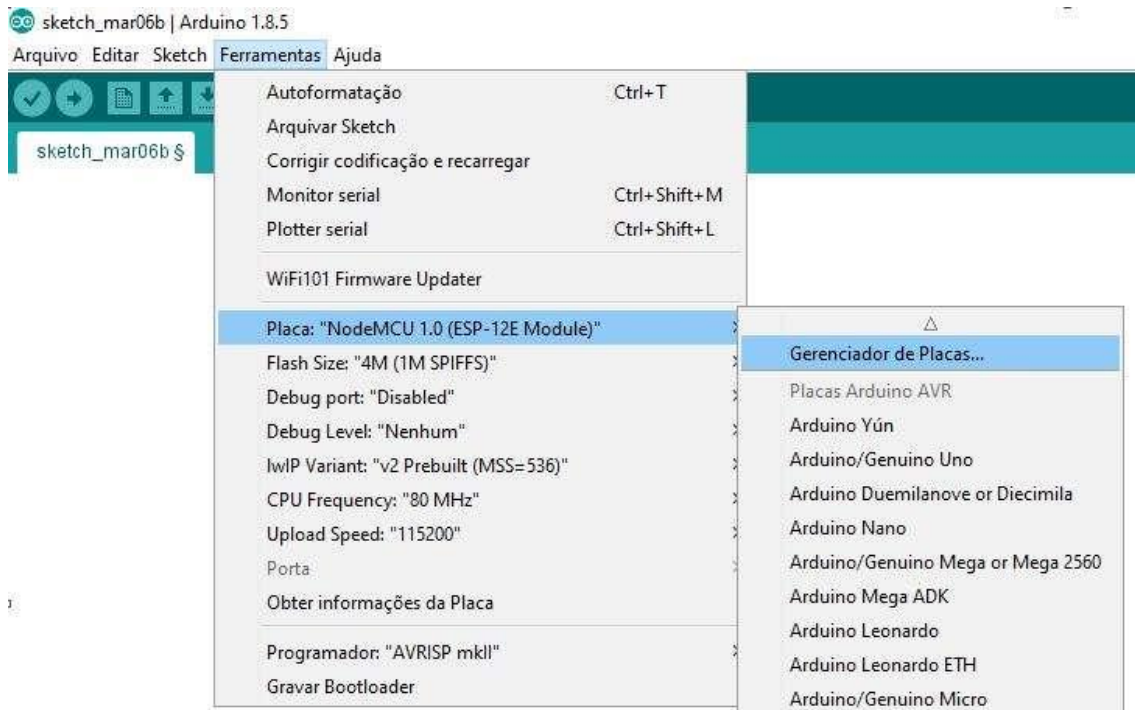
O primeiro passo é atualizar a Arduino IDE. Algumas IDEs antigas não suportam o NodeMCU ESP8266, por isso é necessário atualizar para a versão mais recente. Atualmente a versão mais nova é a 1.8.13.

Baixe a Arduino IDE e instale no seu computador Windows. Existem versões para MacOS e Linux também.

- https://www.arduino.cc/en/software

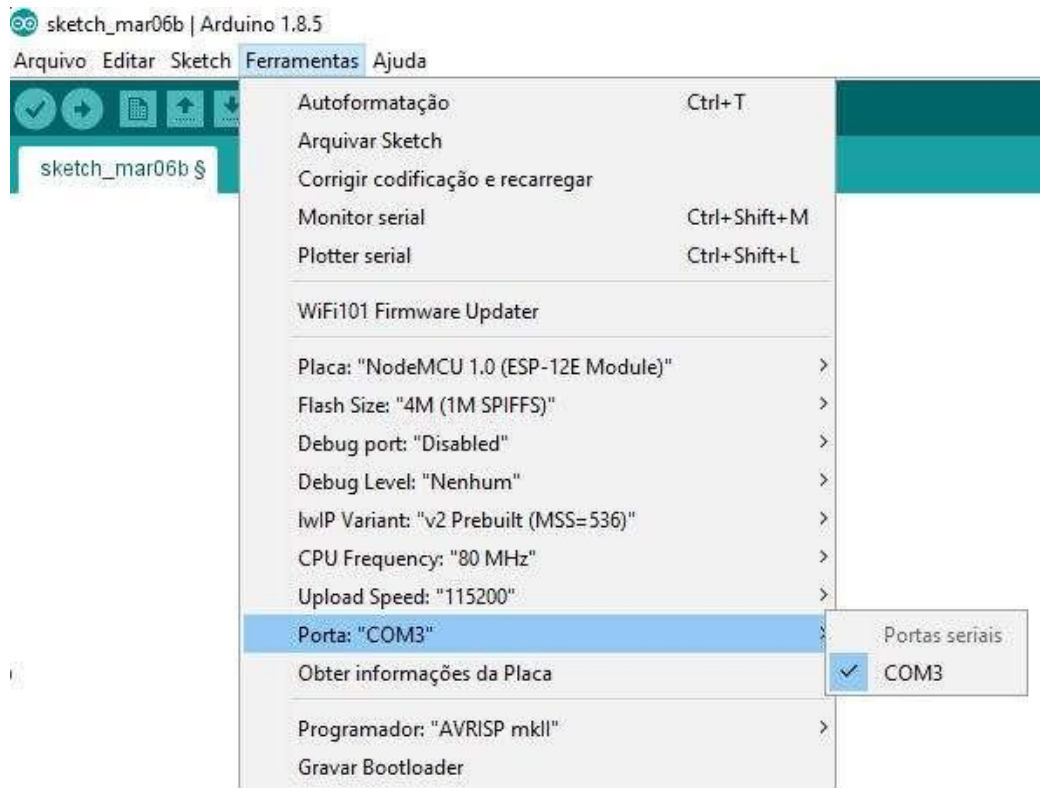
Inicie o programa Arduino IDE. Clique na barra superior em **Arquivos** e depois em **Preferências**.

Dentro da janela de **Preferências**, copie o link abaixo no campo URLs adicionais para **Gerenciadores de Placas** e depois clique em **OK**.

http://arduino.esp8266.com/stable/package_esp8266com_index.json

Na barra superior novamente, clique em **Ferramentas** e depois em **Gerenciamento de Placas**.

Obs: na minha IDE, o ESP8266 já estava instalado.

Na janela do **Gerenciador de Placas**, refine a sua pesquisa com: **ESP8266**. Clique em **More Info** e depois em **Instalar**. Lembre-se que o seu computador deve estar conectado na Internet. Após alguns segundos aparecerá **INSTALLED**. Isto é, instalado. Feche essa janela.

Para a escolher o modelo da sua placa, isto é **NodeMCU ESP12** , clique novamente em **Ferramentas** e depois em **Gerenciamento de Placas**. Na janela das Placas, escolha **NodeMCU 1.0 (ESP12E Module)**. Não altere os outros parâmetros da Placa NodeMCU. Somente altere a porta COM da interface serial-USB. No meu caso a porta é a COM3. O procedimento para identificar qual porta COM é usada no Windows está no Tutorial:

https://blog.eletrogate.com/nodemcu-esp12-introducao-1/

Saiba que a velocidade padrão da Console da IDE Arduino para o ESP8266 é de 115200 Bps.

IMPORTANTE: Após tudo configurado com sucesso, feche e abra o programa Arduino IDE novamente, para que todas as configurações sejam efetivadas. Se não fizer isso, poderá ter problemas ao usá-la.

Programando NodeMCU ESP12 com a Arduino IDE

Vamos começar os testes do ESP8266 com a Arduino IDE do modo mais simples. Usando exemplos do site **Arduino_ESP8266**. Escolhi o Sketch **Scan WIFI** pois com ele podemos testar o circuito WIFI do ESP8266, sem ter que montar nada além do NodeMCU. Recomendo que insira a Placa NodeMCU em um Protoboard, pois assim seus pinos ficarão protegidos (evitando um possível curto-circuito).

- Conecte o NodeMCU com o cabo USB na porta USB do seu Computador PC,
- Abra a Arduino IDE e verifique se a porta COM do NodeMCU foi reconhecida,
- Copie o programa abaixo na Arduino IDE, compile-o e grave-o no ESP8266 (clique no botão Carregar).

OBS: Quando o programa estiver sendo carregado no ESP8266, o led azul ficará piscando.

```
/*
   Pesquisa de Redes WIFI usando o NodeMCU-ESP12
   Apostila Eletrogate - KIT Automação Residencial
   Arduino IDE 1.8.5 - ESP8266
   Gustavo Murta   07/mar/2018
   baseado em
   https://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/scan-class.html
   Blog Eletrogate https://blog.eletrogate.com/nodemcu-esp12-usando-arduino-ide-2/
*/
#include "ESP8266WiFi.h"

void setup(){
 Serial.begin(115200);       // configura monitor serial 115200 Bps
 Serial.println();           // imprime uma linha
 WiFi.mode(WIFI_STA);        // configura rede no modo estacao
 WiFi.disconnect();          // desconecta rede WIFI
 delay(100);                 // atraso de 100 milisegundos
}

void prinScanResult(int networksFound){
 Serial.printf("\n");                                          // imprime uma linha
 Serial.printf("%d redes encontradas\n", networksFound);    // imprime numero de redes
encontradas
 for (int i = 0; i < networksFound; i++){                       // contagem das redes
encontradas
   Serial.printf("%d: %s, Ch:%d (%ddBm) %s\n", i + 1, WiFi.SSID(i).c_str(),
WiFi.channel(i), WiFi.RSSI(i), WiFi.encryptionType(i) == ENC_TYPE_NONE ? "aberta" : "");
 }
}

void loop(){
 WiFi.scanNetworksAsync(prinScanResult);        // imprime o resultado
 delay(500);                                    // atraso de 0,5 segundos
}
```

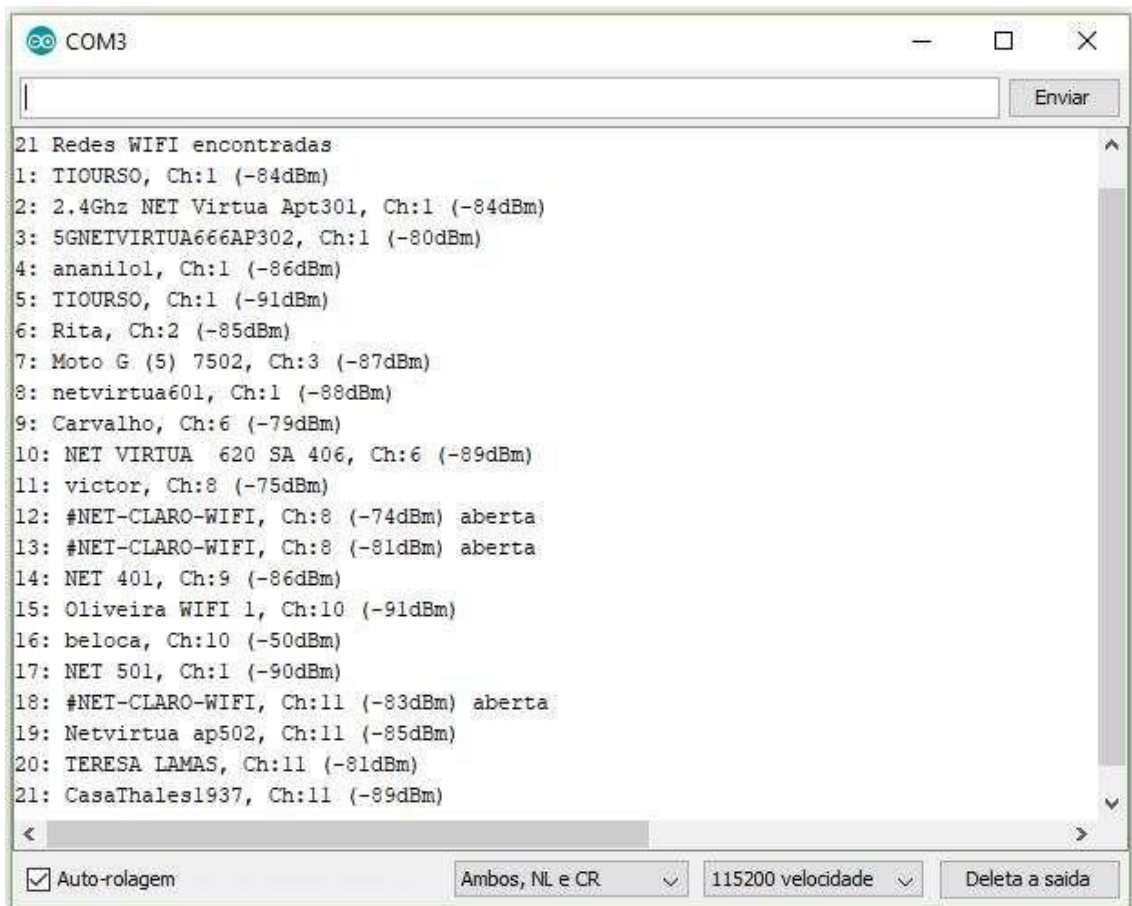
Sketch para pesquisar redes WiFi (próximas ao NodeMCU):

https://github.com/Gustavomurta/NodeMCU-ESP-12-Exemplos/blob/master/ESP8266_ScanWIFI.ino

Console da Arduino IDE – ESP8266 (115200 Bps):

(clique no botão Monitor serial e altere a velocidade para 115200 Bps na barra inferior da janela)

Exemplo – redes encontradas em minha residência .

Nomes das Redes, canal WIFI usado e o nível de transmissão em dBms.

Para entender alguns detalhes do uso do ESP8266 com a Arduino IDE, recomendo que leia essa documentação:

https://arduino-esp8266.readthedocs.io/en/2.4.0/reference.html

Informações sobre as Bibliotecas já instaladas na IDE e mais algumas não instaladas, poderá encontrar no Link:

https://arduino-esp8266.readthedocs.io/en/2.4.0/libraries.html

Sobre a Biblioteca ESP8266WIFI :

https://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/readme.html

Sobre o primeiro Sketch a ser rodado no NodeMCU ESP12 , veja as funções do SCAN WIFI :

https://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/scan-class.html

Parte III - Seção de Exemplos Práticos

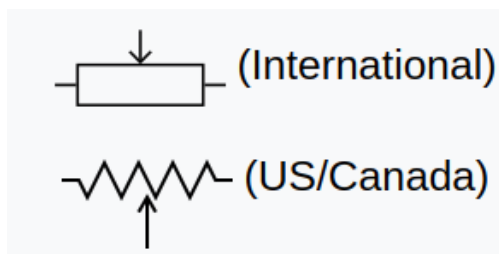
Agora vamos entrar nas seções de exemplos em si. Os conceitos da Parte I são importantes caso você esteja começando a trabalhar com eletrônica. No entanto, devido ao aspecto prático da montagem, não necessariamente você precisa de ler toda a parte introdutória para montar os circuitos abaixo.

Em cada exemplo vamos apresentar os materiais utilizados, o diagrama de circuito e o código base com as devidas explicações e sugestões de referências.

Exemplo 1 - Usando potenciômetro para fazer leituras analógicas

O potenciômetro nada mais é do que um resistor cujo valor de resistência pode ser ajustado de forma manual. Existem potenciômetros slides e giratórios. Na figura abaixo, mostramos um potenciômetro giratório dos mais comumente encontrados no mercado.

Em ambos, ao variar a posição da chave manual, seja ela giratória ou slide, o valor da resistência entre o terminal de saída e um dos outros terminais se altera. O símbolo do potenciômetro é o mostrado na seguinte imagem:

(International)
(US/Canada)

Nesse exemplo, vamos ligar a saída de um potenciômetro a uma entrada analógica do NodeMCU ESP8266. Dessa forma, vamos ler o valor de tensão na saída do potenciômetro e vê-la variando de 0 a 1023. Mas como assim, 0 a 1023?

Isso se deve ao seguinte. Vamos aplicar uma tensão de 3.3V nos terminais do potenciômetro. A entrada analógica do NodeMCU, ao receber um sinal de tensão externo, faz a conversão do mesmo para um valor digital, que é representado por um número inteiro de 0 a 1023. Esses números são assim devido à quantidade de bits que o conversor analógico digital do NodeMCU trabalha, que são 10 bits (2 elevado a 10 = 1024).

Ou seja, o NodeMCU divide o valor de tensão de referência (3.3V) em 1024 unidades (0 a 1023) de 0,00322 volts. Assim, se a tensão lida na entrada analógica for de 2,5V, o valor capturado pelo arduino será metade de 1,65/0,00322 = 512. Se for 0V, será 0, e se for 3.3V, será 1023, e assim proporcionalmente para todos os valores. Assim, digamos que o valor de tensão será dado por V. O valor que o NodeMCU vai te mostrar é:

Valor = (V/3.3) * 1023

Em que 3.3V é o valor de referência configurado no conversor analógico-digital (uma configuração já padrão, não se preocupe com ela) e 1023 é igual 2 elevado a 10 menos 1, uma vez que o conversor começa a contagem a partir de 0.

No nosso código, queremos saber o valor de tensão na saída do potenciômetro, e não esse número de 0 a 1023, assim, rearranjar a equação para o seguinte:

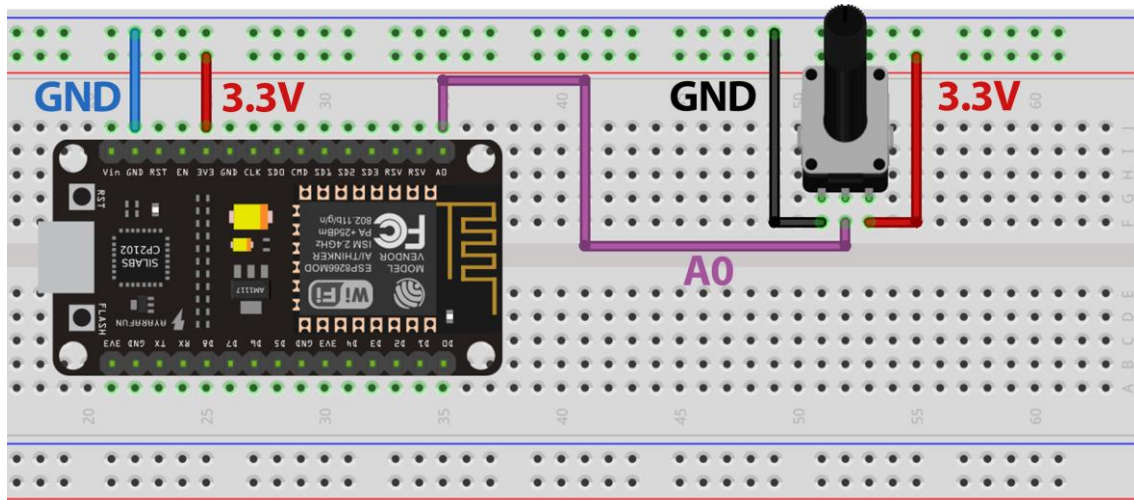
Tensão = Valor*(3.3/1024)

Bacana, né? Agora, vamos à montagem em si.

Lista de materiais:

- NodeMCU ESP8266;
- Protoboard;
- Potenciômetro 10K;
- Jumpers de ligação;

Diagrama de circuito

Monte o circuito conforme diagrama abaixo e carregue o código de exemplo:

O Potenciômetro possui 3 terminais, sendo que o do meio é o que possui resistência variável. A ligação consiste em ligar os dois terminais fixos à uma tensão de 3.3V. Assim, o terminal intermediário do potenciômetro terá um valor que varia de 0 a 3.3V à medida que você gira seu knob.

O terminal intermediário é ligado diretamente a uma entrada analógica do NodeMCU A0. Como a tensão é de no máximo 3.3V, então não há problema em ligar direto. Carregue o código abaixo no NodeMCU e você verá as leituras no Monitor serial da IDE Arduino.

Carregue o código abaixo e observe a medição da Tensão no Monitor Serial da IDE Arduino, ao girar o potenciômetro :

```
// Exemplo 1 - Usando potenciometro para fazer leituras analógicas
// Apostila Eletrogate - KIT Automação Residencial

#define sensorPin A0       // define entrada analógica pino A0

int sensorValue = 0;       // variável inteiro igual a zero
float voltage;             // variável numero fracionario

void setup()
{
  Serial.begin(115200);      // monitor serial - velocidade 9600 Bps
  delay(100);              // atraso de 100 milisegundos
}

void loop()
```

No código acima, nós declaramos a variável **sensorValue** para armazenar as leituras da entrada analógica **(porta)** e a variável do tipo **float Voltage** para armazenar o valor lido convertido para tensão.

Na função **void Setup()**, nós inicializamos o terminal serial com uma taxa de transmissão de 9600 kbps. Na função **void Loop(),** primeiro faz-se a leitura da entrada analógica (porta) com a função **analogRead(SensorPin)** e armazenamos a mesma na variável **sensorValue**. Em seguida, aplicamos a fórmula para converter a leitura (que é um número entre 0 e 1023) para o valor de tensão correspondente. O resultado é armazenado na variável **Voltage** e em seguida mostrado na interface serial da IDE Arduino.

Referência:

- *https://www.arduino.cc/en/Tutorial/ReadAnalogVoltage*

Exemplo 2 - Entradas e saídas digitais - push-button + LED

Os push-buttons (chaves-botões) e LEDs são elementos presentes em praticamente qualquer circuito eletrônico. As chaves são usadas para enviar comandos para o NodeMCU e os LEDs são elementos de sinalização luminosa.

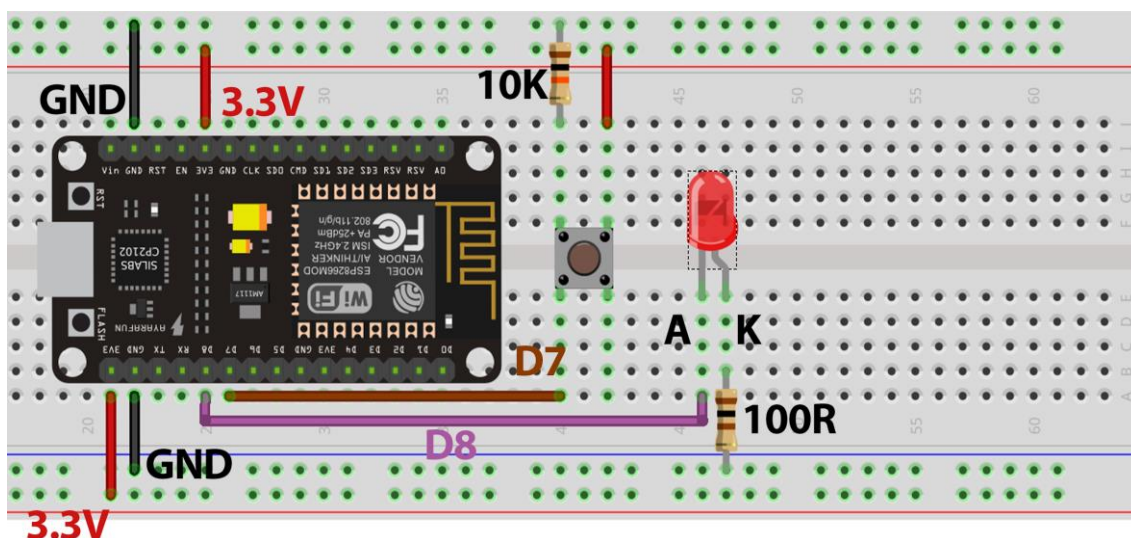
Esses dois componentes são trabalhados por meio das entradas e saídas digitais do NodeMCU. Neste exemplo, vamos fazer uma aplicação basica que você provavelmente

vai repetir muitas vezes. Vamos ler o estado de um push-button e usá-la para acender ou apagar um led. Ou seja, sempre que o botão for acionado, vamos apagar ou desligar o Led.

Lista de materiais:

Para esse exemplo você vai precisar:

- 1 resistor de 100 Ω,
- 1 Led vermelho (ou de outra cor de sua preferência),
- Push-button (chave botão),
- NodeMCU ESP8266,
- Protoboard,
- Jumpers de ligação.

Diagrama de circuito:

Esse pequeno código abaixo mostra como ler entradas digitais e como acionar as saídas digitais. Na função **void Setup(),** é preciso configurar qual pino será usado como saída e qual será usado como entrada.

Depois de configurar os pinos, para acioná-los basta chamar a função **digitalWrite(pino,HIGH).** A função **digitalWrite()** aciona ou desativa um pino digital dependendo do valor passado no argumento. Se for "HIGH", o pino é acionado. Se for "LOW", o pino é desligado.

Na função **void Loop(),** fizemos um if no qual a função **digitalRead** é usada para saber se o pushButton está acionado ou não. Caso ele esteja acionado, nós acendemos o Led, caso ele esteja desligado, nós desligamos o led.

Carregue o código abaixo e pressione o botão para acender o LED.

```
// Exemplo 2 - Entradas e saídas digitais - push-button + led
// Apostila Eletrogate - KIT Automação Residencial

#define PinButton 13                        // define pino digital D7
#define ledPin 15                           // define pino digital D8

void setup(){
  pinMode(PinButton, INPUT);               // configura D7 como entrada digital
  pinMode(ledPin, OUTPUT);                 // configura D8 como saída digital
  digitalWrite(ledPin, LOW);               //Iniciando o LED desligado
  Serial.begin(115200);                    // monitor serial - velocidade 9600 Bps
  delay(100);                              // atraso de 100 milisegundos
}

void loop(){
  if ( digitalRead(PinButton) == LOW){      // se botão = nível baixo
    digitalWrite(ledPin, LOW);              // desliga LED
    Serial.print("Desligando Led");          // imprime no monitor serial
  }
  else{                                     // senão botão = nível alto
    digitalWrite(ledPin, HIGH);             // desliga LED
    Serial.print("Acendendo led");         // imprime no monitor serial
  }
  delay(100);                               // atraso de 100 milisegundos
}
```

Referência:

- *https://www.arduino.cc/reference/en/language/func0tions/digital-io/digitalread/*

Exemplo 3 - Sensor de luz LDR

O sensor LDR é um sensor de luminosidade. LDR é um **Light Dependent Resistor**, ou seja, um resistor cuja resistência varia com a quantidade de luz que incide sobre ele. Esse é seu princípio de funcionamento.

Quanto maior a luminosidade em um ambiente, menor será a resistência do LDR. Essa variação na resistência é medida através da queda de tensão no sensor, que varia

proporcionalmente (de acordo com a lei Ohm, lembra?) com a queda na resistência elétrica.

A imagem abaixo mostra o sensor em mais detalhes:

Fotoresistor (LDR)

É importante considerar a potência máxima do sensor, que é de 100 mW. Ou seja, com uma tensão de operação de 5V, a corrente máxima que pode passar por ele é 20 mA. Felizmente, com 8 KΩ (que medimos experimentalmente com o ambiente bem iluminado), que é a resistência mínima, a corrente ainda está longe disso, sendo 0,625mA. Dessa forma, podemos conectar o sensor diretamente com o NodeMCU.

**Nota: Nas suas medições, pode ser que você encontre um valor de resistência mínimo diferente, pois depende da iluminação local.*

Especificações do LDR:

- Modelo: GL5528
- Diâmetro: 5mm
- Tensão máxima: 150VDC
- Potência máxima: 100mW
- Temperatura de operação: -30°C a 70°C
- Comprimento com terminais: 32mm
- Resistência no escuro: 1 MΩ (Lux 0)
- Resistência na luz: 10-20 KΩ (Lux 10)

Este sensor de luminosidade pode ser utilizado em projetos com Arduino e outros microcontroladores para alarmes, automação residencial, sensores de presença e vários outros.

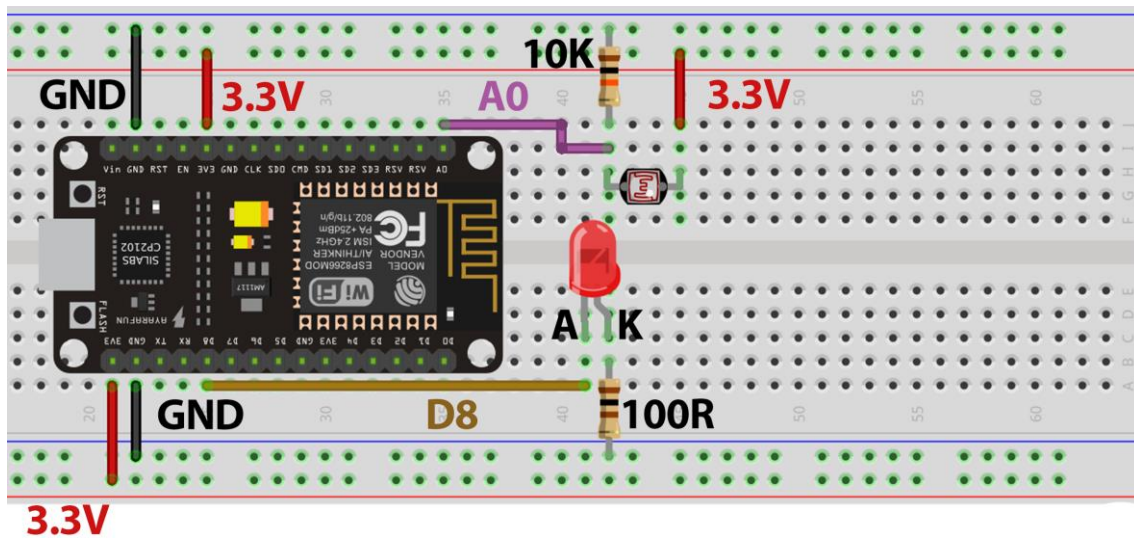
Nesse exemplo, vamos usar uma entrada analógica do NodeMCU ESP8266 para ler a variação de tensão no LDR e, consequentemente, saber como a luminosidade ambiente está se comportando. Veja na especificação que com muita luz, a resistência fica em torno de 10-20 KΩ, enquanto que no escuro pode chegar a 1MΩ.

Para podermos ler as variações de tensão resultantes da variação da resistência do LDR, vamos usar o sensor como parte de um divisor de tensão. Assim, a saída do divisor será dependente apenas da resistência do sensor, pois a tensão de entrada e a outra resistência são valores conhecidos. No nosso caso, vamos usar um resistor de 10K e uma tensão de operação de 5V. Assim, o sinal que vamos ler no arduino terá uma variação de 2,2V (quando o LDR for 8K) e 5V (quando o LDR tiver resistências muito maiores que o resistor de 10K).

Lista de materiais:

Para esse exemplo você vai precisar:

- LDR;
- 1 Resistor de 10 KΩ e 1 Resistor de 100 Ω;
- NodeMCU ESP8266;
- Protoboard;
- Jumpers de ligação;

Diagrama de circuito:

Carregue o código abaixo e varie a luminosidade sobre o LDR.

```
// Exemplo 3 - Sensor de luz LDR
// Apostila Eletrogate - KIT Automação Residencial

#define AnalogLDR A0                       // define pino analógico A0
#define Limiar 1.5                         // define constante igual a 1.5
#define ledPin 15                           // define pino digital D8

int Leitura = 0;                           // variavel inteiro igual a zero
float VoltageLDR;                          // variável número fracionário
float ResLDR;                              // variável número fracionário

void setup(){
  pinMode(ledPin, OUTPUT);                   // configura D15 como saída digital
  Serial.begin(115200);                        // monitor serial - velocidade 9600
Bps
  delay(100);                                // atraso de 100 milissegundos
}

void loop(){
  Leitura = analogRead(AnalogLDR);           // leitura da tensão no pino analogico
A0
  VoltageLDR =  Leitura * (3.3/1024);        // cálculo da tensão no LDR
  Serial.print("Leitura sensor LDR = ");     // imprime no monitor serial
  Serial.println(VoltageLDR);                // imprime a tensão do LDR

  if (VoltageLDR > Limiar)                   // se a tensão LDR maior do que limiar
    digitalWrite(ledPin, HIGH);              // liga LED com 3.3V
  else                                       // senão a tensão LDR < limiar
    digitalWrite(ledPin, LOW);               // desliga LED com 0V
  delay(500);                                // atraso de 500 milisegundos
}
```

Na montagem, o sensor é ligado como parte de um divisor de tensão no pino analógico A0, de forma que a tensão de saída do divisor varia de acordo com a variação da resistência do sensor LDR. Assim, vamos identificar as variações na intensidade de luz

pelas variações na tensão do sensor. Quanto maior a intensidade de luz, menor será a resistência do sensor e, consequentemente, menor a tensão de saída.

No código acima, usamos funcionalidades de todos os exemplos anteriores. Como o sensor é um elemento de um divisor de tensão, fazemos a sua leitura do mesmo modo que nos exemplos do potenciômetro e divisor de tensão.

Nesse caso, definimos uma tensão de limiar, a partir da qual desligamos ou ligamos um led para indicar que a intensidade da luz ultrapassou determinado valor. Esse limiar pode ser ajustado por você, para desligar o led com intensidades diferentes de luz ambiente.

É importante que o sensor esteja exposto à iluminação ambiente e não sofra interferência de fontes luminosas próximas, mas que não sejam parte do ambiente.

Referências:

- *https://maker.pro/education/using-an-ldr-sensor-with-arduino-a-tutorial-for-beginners*
- *http://blog.eletrogate.com/controle-de-luminosidade-com-arduino-e-sensor-ldr/*

Exemplo 4 - Sensor de temperatura NTC

O sensor de temperatura NTC pertence a uma classe de sensores chamada de Termistores. São componentes cuja resistência é dependente da temperatura. Para cada valor de temperatura absoluta há um valor de resistência.

Assim, o princípio para medir o sinal de um termistor é o mesmo que usamos para medir o sinal do LDR. Vamos medir, na verdade, a queda de tensão provocada na resistência do sensor.

Existem dois tipos básicos de termistores:

- **PTC (Positive Temperature Coefficient):** Nesse sensor, a resistência elétrica aumenta à medida que a temperatura aumenta;
- **NTC (Negative Temperature Coefficient):** Nesse sensor, a resistência elétrica diminui à medida que a temperatura aumenta.

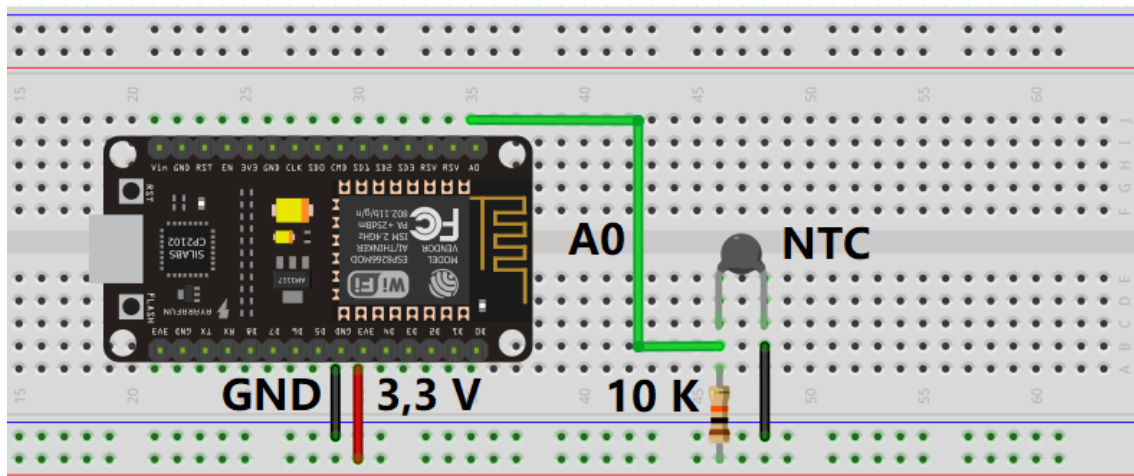
O termistor que acompanha o kit é do tipo NTC, como o próprio nome diz. Esse é o tipo mais comum do mercado.

Para usarmos os termistores é necessário fazer uma função de calibração, pois a relação entre temperatura e resistência elétrica não é linear. Existe uma equação, chamada equação de **Steinhart-Hart,** que é usada para descrever essa relação.
O código que vamos usar nesse exemplo utiliza a equação de Steinhart-Hart conforme essa referência:

http://www.instructables.com/id/NTC-Temperature-Sensor-With-Arduino/

Lista de materiais:

- 1 Sensor de temperatura NTC,
- NodeMCU ESP8266,
- 1 resistor de 10 KΩ,
- Protoboard;
- Jumpers para conexão no protoboard;

Diagrama de circuito:

Para uma melhor precisão nas medidas de temperatura, meça a tensão de 3,3V no pino 3V3 do NodeMCU, usando um multímetro (não contém no KIT). Essa tensão é a usada como referência do conversor analógico-digital do NodeMCU. Especifique esse valor de tensão na primeira linha do Sketch:

```
#define Vin 3.3
```

Por exemplo, se a tensão for 3.2 V:

```
#define Vin 3.2
```

Carregue o código abaixo e meça a temperatura com o NTC:

```
// Exemplo 4 - Sensor de temperatura NTC
// Apostila Eletrogate - KIT Automação Residencial

#define Vin 3.3                   // define constante igual a 3.3
#define T0 298.15                 // define constante igual a 298.15 Kelvin
#define Rt 10000                  // Resistor do divisor de tensao
#define R0 10000                  // Valor da resistencia inicial do NTC
#define T1 273.15                 // [K] in datasheet 0º C
#define T2 373.15                 // [K] in datasheet 100° C
#define RT1 35563                 // [ohms] resistencia in T1
#define RT2 549                   // [ohms] resistencia in T2
float beta = 0.0;                 // parametros iniciais [K]
float Rinf = 0.0;                 // parametros iniciais [ohm]
float TempKelvin = 0.0;           // variable output
float TempCelsius = 0.0;          // variable output
float Vout = 0.0;                 // Vout in A0
float Rout = 0.0;                 // Rout in A0

void setup(){
  Serial.begin(115200);                       // monitor serial
  beta = (log(RT1 / RT2)) / ((1 / T1) - (1 / T2));     // calculo de beta
  Rinf = R0 * exp(-beta / T0);              // calculo de Rinf
  delay(100);                               // atraso de 100 milisegundos
}

void loop(){
  Vout = Vin * ((float)(analogRead(0)) / 1024.0); // calculo de V0 e leitura de A0
  Rout = (Rt * Vout / (Vin - Vout));              // calculo de Rout
  TempKelvin = (beta / log(Rout / Rinf));         // calculo da temp. em Kelvins
  TempCelsius = TempKelvin - 273.15;              // calculo da temp. em Celsius
  Serial.print("Temperatura em Celsius: ");       // imprime no monitor serial
  Serial.print(TempCelsius);                      // imprime temperatura Celsius
  Serial.print("   Temperatura em Kelvin: ");     // imprime no monitor serial
  Serial.println(TempKelvin);                     // imprime temperatura Kelvins
  delay(500);                                     // atraso de 500 milisegundos
}
```

Referências e sugestões de leitura:

- https://www.ametherm.com/thermistor/ntc-thermistors-steinhart-and-hart-equation
- https://www.thinksrs.com/downloads/pdfs/applicationnotes/LDC%20Note%204%20NTC%20Calculator.pdf
- https://www.mundodaeletrica.com.br/sensor-de-temperatura-ntc-ptc/

Exemplo 5 - Módulo Relé 4 Canais

O Módulo Relé nada mais é do que um ou mais relés acionados por sinais digitais de 5V. Esse componente é indicado para acionar cargas que utilizam correntes contínuas maiores do que a suportada pelo NodeMCU, ou então uma carga de um circuito de corrente alternada (rede elétrica).

O módulo de relé pode ser usado para vários tipos de aplicações:

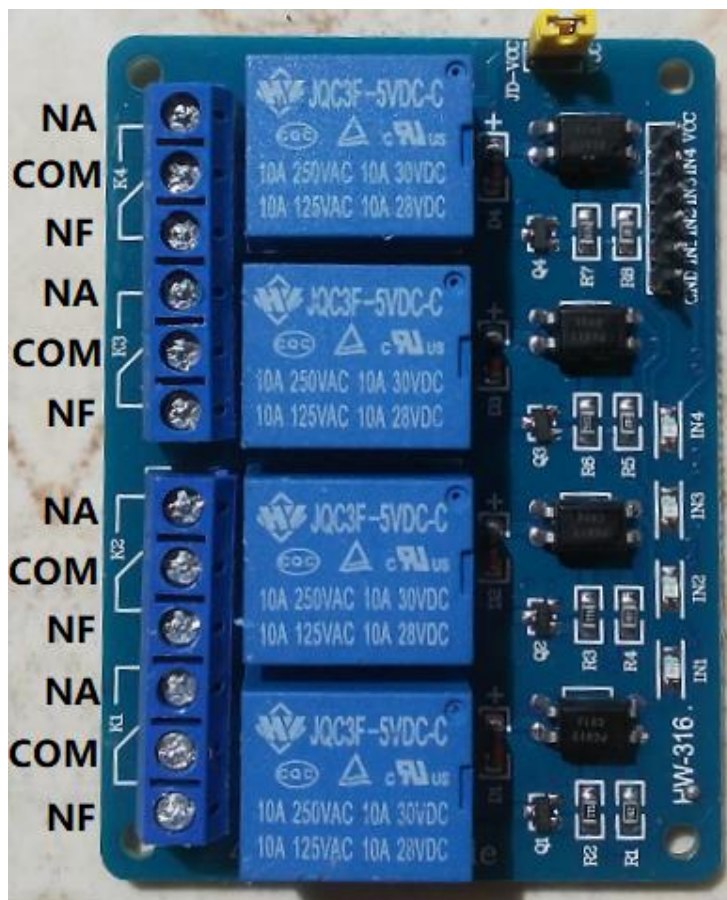
- Ativação de eletroímãs,
- Ativação de motores CC,
- Ativação de motores CA,
- Ligar/desligar lâmpadas.

No mercado, você pode encontrar módulos de 1, 2, 4 e 8 canais. Cada canal corresponde a um relé montado no módulo. Ou seja, um módulo relé de 5V e 8 canais, é um conjunto de 8 relés acionados por sinais separados de 5V. Um módulo relé de 12V e 4 canais, nada mais do que um conjunto de 4 relés, cada qual acionado por sinais de 12V.

O módulo com 4 relés tem um circuito de controle através de acopladores óticos. Isso é usado para isolar as portas do microcontrolador, do circuito de acionamento dos relés.

As portas de controle dos relés são os pinos IN1, IN2, IN3 e IN4. Por exemplo, para acionar o relé K1, a porta do microcontrolador conectada no pino IN1 deverá estar no nível baixo (LOW). Para controlar o relé K2, use o pino IN2, e assim por diante.

Existe um Led em série com cada entrada do acoplador óptico. Portanto, quando a porta é ativada, o Led acende. A alimentação do módulo tem que ser 5 V (pino +5V). Não use uma tensão maior ou menor do que 5 V, pois poderá queimar o relé (tensão maior) ou então o relé não poderá ser acionado (tensão menor). Se quiser usar relés com 12 V, procure um outro módulo de relé. Não se esqueça de conectar o pino terra (GND) do módulo com o GND do NodeMCU. Mantenha o jumper entre os pinos JD-VCC e VCC. Dessa forma, a alimentação dos relés será de 5 V através do pino +5V. A capacidade corrente em cada contato é de 10 A. Não ultrapasse esse valor, para não danificar o seu relé. O consumo de corrente da bobina de cada relé é de aproximadamente 70 mA, quando energizada.

O módulo funciona exatamente da mesma forma que uma chave ou interruptor. Na imagem abaixo, temos o Módulo Relé 4 canais com a identificação das conexões:

- NA - Contato Normalmente Aberto
- COM - Terminal Comum
- NF - Contato Normalmente Fechado

Os pinos dos contatos dos Relés são NA (normalmente aberto), COM (comum) e NF (normalmente fechado). Isto é, enquanto o relé estiver desativado, a ponta comum estará conectada no contato NF. Quando o relé for acionado, a ponta comum será conectada no contato NA.

Cada pino **INX** serve para acionar um dos relés. **VCC** e **GND** são os pinos de alimentação do Módulo do relé: VCC deve ser conectado no +5V e o GND no pino terra, ambos da fonte. **O pino terra (GND) do Módulo Relé deve estar conectado também no terra (GND) do Arduino!**

Lista de Materiais:

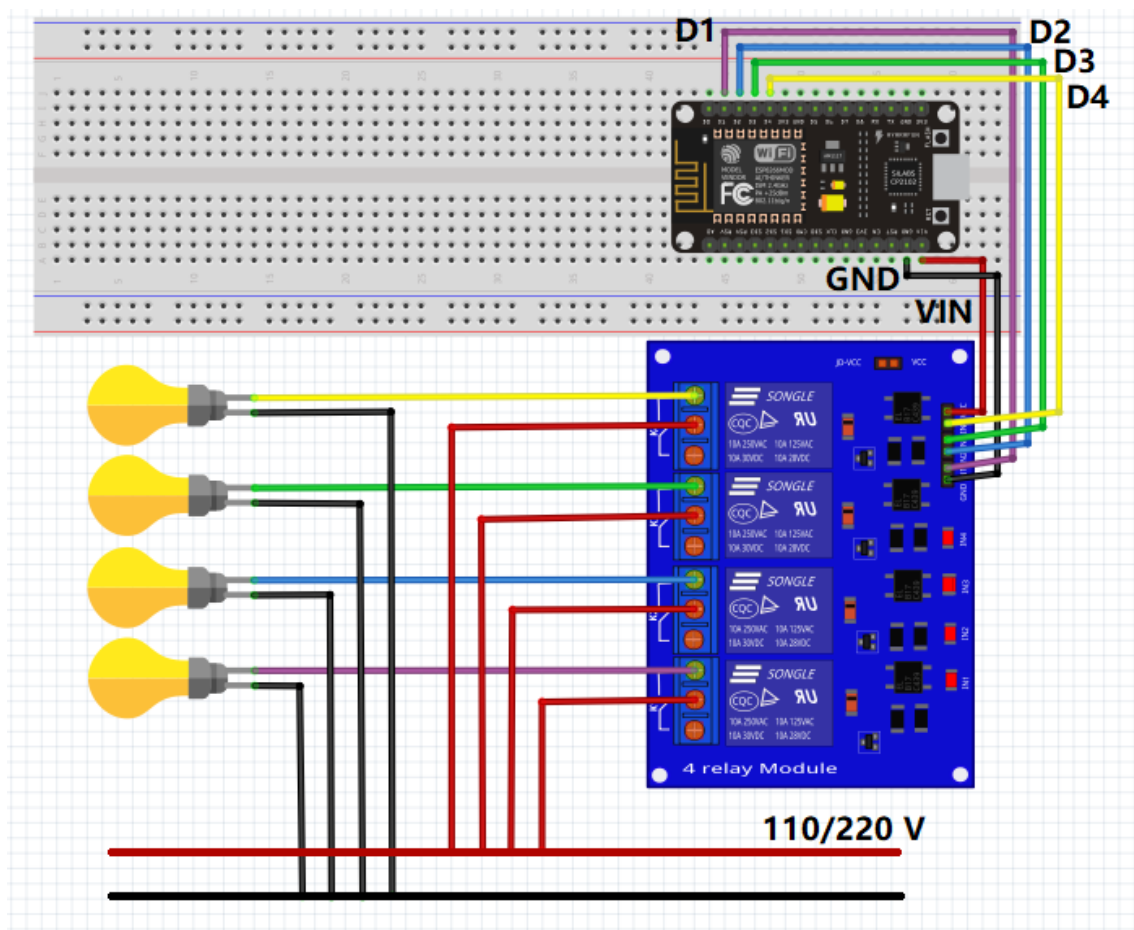
Para este exemplo você vai precisar:

- Protoboard,
- NodeMCU ESP8266,

- Jumpers de ligação,
- Módulo relé 4 canais,
- Lâmpadas e cabos elétricos.

O circuito abaixo é uma montagem simples de acender lâmpadas usando relés. Quatro lâmpadas foram conectadas aos quatro terminais do relé, os quais serão acionados pelo NodeMCU. Tanto o código como a montagem são de fácil entendimento e confecção, já que esse exemplo tem o único objetivo de familiarizar o leitor com a utilização do relé.

É interessante notar que, diferentemente dos outros exemplos, dessa vez a fonte de entrada usada do NodeMCU foi o Vin, e não o 3,3 V, já que o relé só trabalha com 5 V. As conexões são bastante intuitivas, conecte o GND do NodeMCU com o GND do relé, o Vin do NodeMCU com o VCC do relé, e as portas D1, D2, D3 e D4 do NodeMCU com as portas IN1, IN2, IN3 e IN4 do relé, respectivamente. A ligação da lâmpada com o relé será feita da seguinte forma: a porta normalmente aberta do relé será conectado a uma das entradas da lâmpada, enquanto a porta comum do relé e a outra conexão da lâmpada serão conectados a uma alimentação externa, como uma tomada, por exemplo.

Diagrama do circuito:

Basicamente, cada luz irá acender a cada 3 segundos e, posteriormente, todas irão se apagar, e o processo se repete infinitamente.

**Nota: Se for menor de idade, peça ajuda a um adulto para fazer as ligações elétricas.*

Código do Projeto:

```
// Exemplo 5 - Módulo 4 relés
// Apostila Eletrogate - Automação Residencial

int IN1 = 5;                              // pino IN1 conectado porta D1 NodeMCU
int IN2 = 4;                              // pino IN2 conectado porta D2 NodeMCU
int IN3 = 0;                              // pino IN3 conectado porta D3 NodeMCU
int IN4 = 2;                              // pino IN4 conectado porta D4 NodeMCU

void setup()
{
  Serial.begin(115200);                   // console serial 115200 Bps
  pinMode(IN1, OUTPUT);                   // definições das portas como portas de saídas
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}

void loop()
{
  digitalWrite(IN1, LOW);                // acionamento relé 1
  Serial.println("Lâmpada 1 ligada");     // print mensagem
  delay (3000);                           // atraso de 3 segundos
  digitalWrite(IN2, LOW);                // acionamento relé 2
  Serial.println("Lâmpada 2 ligada");     // print mensagem
  delay (3000);                           // atraso de 3 segundos
  digitalWrite(IN3, LOW);                // acionamento relé 3
  Serial.println("Lâmpada 3 ligada");     // print mensagem
  delay (3000);                           // atraso de 3 segundos
  digitalWrite(IN4, LOW);                // acionamento relé 4
  Serial.println("Lâmpada 4 ligada");     // print mensagem
  delay (3000);                           // atraso de 3 segundos
  digitalWrite(IN1, HIGH);                 // desliga lâmpada 1
  digitalWrite(IN2, HIGH);                 // desliga lâmpada 2
  digitalWrite(IN3, HIGH);                 // desliga lâmpada 3
  digitalWrite(IN4, HIGH);                 // desliga lâmpada 4
  Serial.println("Lâmpadas desligadas"); // print mensagem
  delay (3000);                           // atraso de 3 segundos
}
```

- https://blog.eletrogate.com/flip-flop-d-ligando-e-desligando-rele-com-pushbutton/
- https://www.allaboutcircuits.com/projects/use-relays-to-control-high-voltage-circuitswwith-an-arduino/
- https://www.arduino.cc/en/Tutorial/Debounce

Exemplo 6 - Display LCD 16x2

O display 16×2 é um dispositivo que possui interface para comunicação e controle padronizada. Esses displays são fáceis de serem controlados e graças a essa padronização é possível trocar um display 16x2 de um fabricante por outro diferente sem maiores problemas.

A tecnologia utilizada nos displays tradicionais é LCD (Liquid Crystal Display). Mais recentemente, modelos mais modernos com tecnologia OLEDs, os LEDS orgânicos, já estão sendo encontrados também.

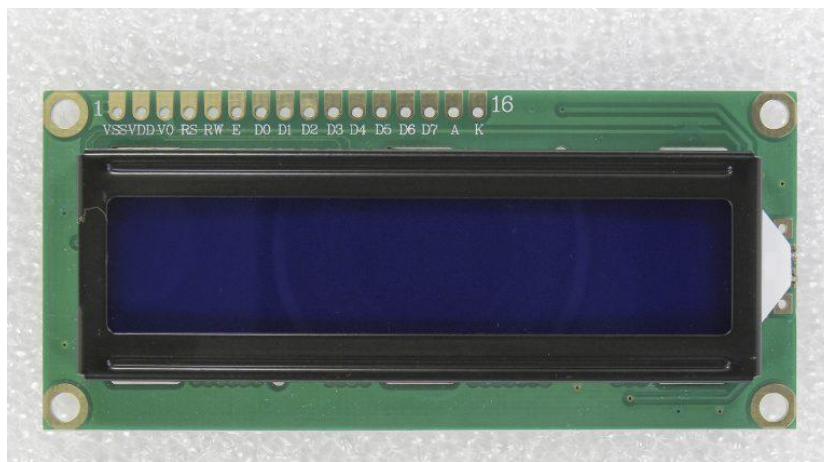
A grande vantagem, e razão, pela qual até hoje é componente popular, é o seu preço e a facilidade de implementação em projetos eletrônicos. Antes dos displays LCD, a interface gráfica usada para mostrar caracteres alfa numéricos era os displays a LED de x segmentos (os mais comuns eram de 7, 14 ou 16 segmentos).

Esses componentes mais antigos utilizavam vários LEDs (7, 14 ou 16) dispostos em segmentos, de forma que dependendo de qual conjunto de LEDs fosse aceso, um determinado caractere seria mostrado.

LCD 16x2. Créditos: Eletrogate

Na hora de comprar um display LCD, você vai encontrar diversas opções, além do mais tradicional 16×2. A especificação é dada em relação ao número de caracteres por linha e o número de linhas. Assim, 16×2 significa que o display pode exibir 16 caracteres em cada linha, e possui 2 linhas. Alguns valores típicos para essas especificações são:

- Quantidade de caracteres padrão de mercado: 8, 12, 16, 20, 24 e 40;
- Quantidade de linhas mais comuns: 1, 2 e 4;
- Cores de fundo (backlight): Azul ou verde;

O display que vamos trabalhar é de 16 colunas e 2 linhas (16×2) e com backlight azul e caracteres brancos. A interface com Arduino é feita por meio de 16 pinos. Em geral apenas 12 são utilizados de fato. Os pinos do LCD são:

LCD 16x2 pinagem – foto Gustavo Murta

- Pino de **register select (RS):** Controle em que parte da memória (do LCD) o usuário está escrevendo os dados. São duas opções, você pode escrever um dado para ser mostrado no display, ou uma instrução no MCU do display;
- Pino de **Read/Write (R/W**): Seleciona se está em modo de leitura ou de escrita;
- Pino de **Enable**: Habilita a escrita de dados nos registradores;
- 8 pinos de dados **(D0 -D7).** Os estados de cada pino correspondem aos bits que você está escrevendo em um registrador do MCU (do display), ou os valores que você lê de um registrador quando está no modo de leitura.
- Pinos de alimentação **VCC e GND** do LCD e do LED Backlight (A-ânodo/K-cátodo);
- Pino **V0** de ajuste de contraste;

Vamos conferir em nosso exemplo como usar o LCD 16x2. Na figura abaixo, mostramos um outro modelo, um display gráfico de 128 colunas e 64 linhas, também muito comercializado e que você pode obter separadamente do Kit para implementar mensagens gráficas mais avançadas.

Display gráfico 128x64

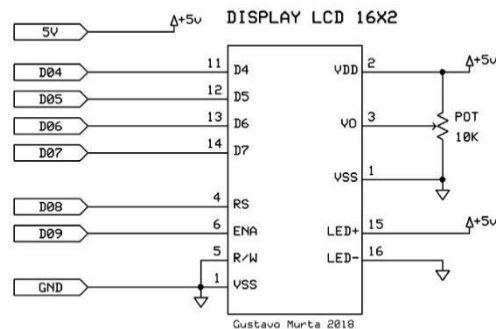
Lista de materiais:

A lista de materiais é a seguinte:

- NodeMCU ESP8266,
- Protoboard,
- Jumpers para ligação no protoboard,
- Display LCD 16x2,
- Potenciômetro 10K.

Diagrama de circuito:

No diagrama acima, o potenciômetro é usado para controlar o contraste do LCD, e deve ser ajustado sempre que você ligar o display para poder visualizar bem cada caracter. As demais ligações são padrão. No código da próxima seção, os pinos do Arduino são configurados de acordo com a ligação acima.

Veja o diagrama eletrônico da montagem acima:

Código:

```
// Exemplo 6 - Display LCD 16x2
// Apostila Eletrogate - KIT Automação Residencial

#include <LiquidCrystal.h>              // biblioteca Liquid Crystal

LiquidCrystal lcd(2, 0, 15, 13, 12, 14); // pinos do LCD - D4 D3 D8 D7 D6 D5
int contador = 0;                        // variável contador

void setup(){
  lcd.begin(16, 2);                      // inicializa LCD 16x2
  delay(500);                            // atraso de 0,5 segundos
}

void loop(){
  lcd.clear();                           // limpa tela do LCD
  lcd.setCursor(0, 0);                   // selecionando coluna 0 e linha 0
  lcd.print("Exemplo LCD !");            // mostra no LCD
  lcd.setCursor(1, 1);                   // selecionando coluna 1 e linha 1
  lcd.print(contador);                   // mostra no LCD a contagem
  contador++;                            // incrementa contador
  if (contador == 60)                    // se contador = 60
    contador = 0;                        // zera o contador
  delay(1000);                           // atraso de 1 segundo
}
```

No código, é mostrado na primeira linha (0) do display, a mensagem “Exemplo LCD !“. Na segunda linha (1) será mostrado um contador que é incrementado de 1 em 1 segundo. Para usar o LCD é preciso incluir a biblioteca **LiquidCrystal.h**, que é nativa da IDE arduino. Essa biblioteca possui as seguintes funções:

- lcd.clear() - Limpa a tela do display,
- lcd.SetCursor(0,0) - Posiciona o cursor a partir de onde os caracteres serão escritos,
- lcd.print() - Imprime caracteres no display.

No exemplo são usadas essas três funções, mas a biblioteca dispõe de várias outras que você pode conferir na página oficial (veja as referências abaixo).

Referências:

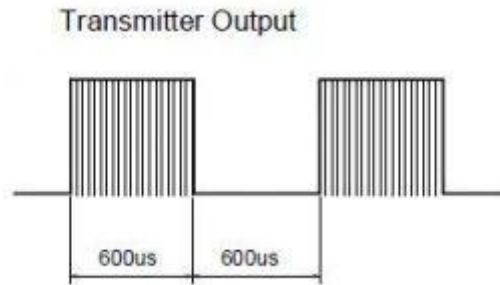
- *http://blog.eletrogate.com/guia-completo-do-display-lcd-arduino/*
- *http://blog.eletrogate.com/display-lcd-modulo-rf-para-deteccao-de-presenca/*
- *https://www.arduino.cc/en/Reference/LiquidCrystal*

Exemplo 7 – Controle Remoto Ir + Receptor Universal Ir

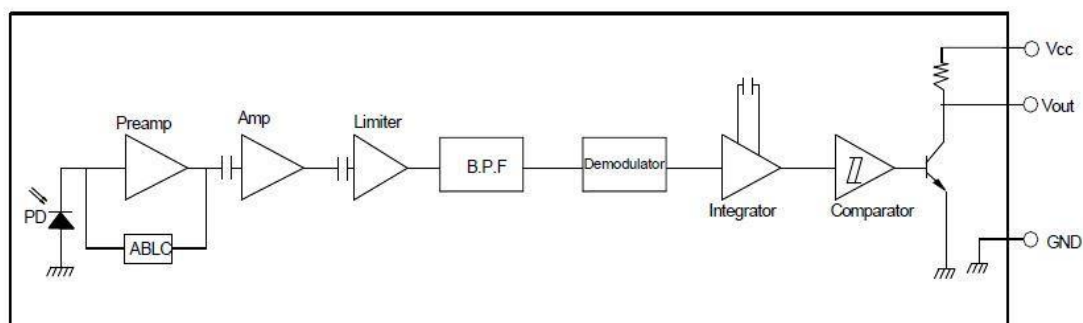
O controle Remoto IR (infrared, ou infravermelho em português) de aparelhos eletrônicos consiste em um pequeno dispositivo que contém um chip de microcontrolador, um ou mais LEDs emissores de infravermelho e um teclado acoplado. Quando o usuário pressiona uma das teclas do controle, uma sequência de pulsos de luz infravermelha é transmitida pelos LEDs. Esses pulsos formam um código que é único para cada tecla acionada.

O chip do Microcontrolador do Controle Remoto Transmissor já vem programado com um protocolo definido pelo fabricante do aparelho.

Os pulsos de dados (código) são enviados do transmissor através de outros pulsos numa frequência bem maior, modulando o sinal a ser transmitido. Alguns trabalham com frequências diferentes de modulação, mas a mais comum atualmente é a frequência de 38 KHz. Na figura abaixo, pode-se perceber que quando o pulso de dados é 1, a frequência é transmitida e quando o pulso é 0, nenhuma frequência é transmitida. Esse tipo de modulação chama-se PCM (modulação codificada de pulsos).

Com esse tipo de modulação, impede-se que a luz externa interfira na transmissão dos dados.

O circuito Receptor IR que fica internamente no aparelho a ser controlado consiste basicamente de um descodificador dos pulsos recebidos do transmissor. Interpretando o código de cada tecla pressionada, o aparelho pode executar vários tipos de operações, como por exemplo, desligar, aumentar volume e mudar o canal no caso de uma Televisão.

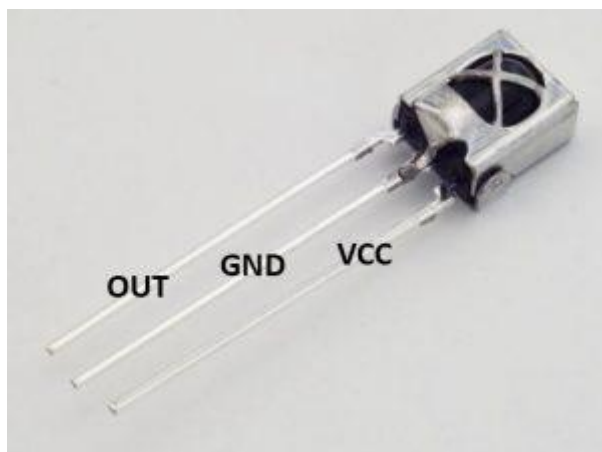
Hoje, com a integração cada vez maior dos circuitos, existem receptores de controle remoto IR que já contém o foto-sensor e todos os outros circuitos necessários para demodular os pulsos de controle, como amplificadores de sinal, filtros, demodulador, integrador e comparador.

O diagrama de blocos abaixo contém os vários circuitos que estão integrados no pequeno receptor. Após a demodulação dos pulsos, esses são decodificados por um outro Microcontrolador, para que o aparelho possa entender que tipo de operação deverá ser realizada.

Receptor de Infravermelho AX-1838HS

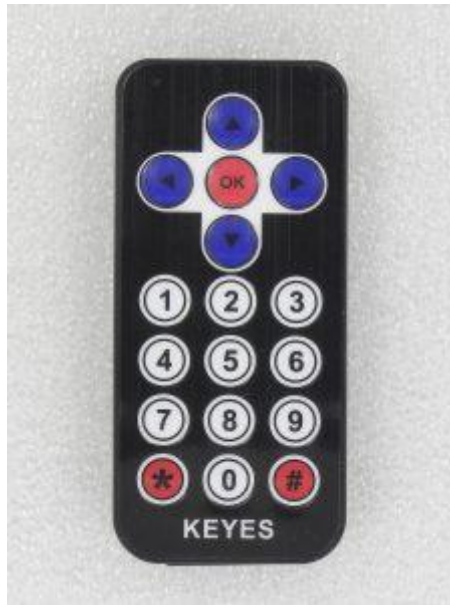
Esse é o foto-receptor IR usado neste exemplo. Ele é amplamente utilizado em aparelhos para permitir o controle remoto. Pode ser encontrado em TVs, rádios, aparelhos de multimídia e até em aparelhos de ar condicionado.

Ele pode trabalhar com tensões de 2,1V a 5,5V. O consumo de energia é bem baixo, com uma corrente de aproximadamente 1,5 mA. A frequência de modulação dos pulsos de dados é de 38 KHz. E o comprimento de onda da luz infravermelha percebida é de 940 nm. O ângulo de visão é de aproximadamente 90 graus. A vantagem do uso desse foto-receptor IR é que ele já tem todo o circuito integrado necessário para demodular os pulsos.

Folha de especificações (Datasheet) do AX-1838HS:

http://blog.eletrogate.com/wp-content/uploads/2018/12/IR-Receiver-AX-1838HS.pdf

Sensor AX-1838HS - pinagem

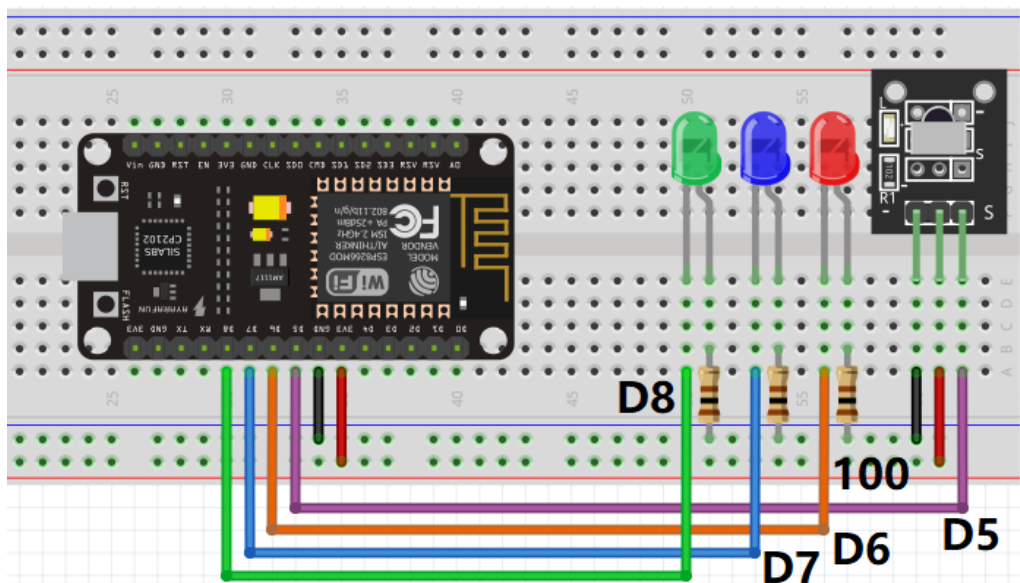
Controle Remoto Infravermelho:

O controle remoto IR (transmissor) é o da KEYES com 17 botões. Ele funciona com uma pilha botão CR2025. Antes de usá-lo, retire a proteção de plástico transparente que fica dentro do suporte da bateria. Para trocar a pilha, pressione a trava para o lado e puxe o suporte da pilha. Veja o diagrama na parte debaixo do controle remoto. Esse controle remoto usa o protocolo de codificação da NEC.

Controle Remoto IR com Arduino:

Usando um circuito de Controle Remoto IR com o NodeMCU ESP8266, poderá implementar inúmeras aplicações bem interessantes como:

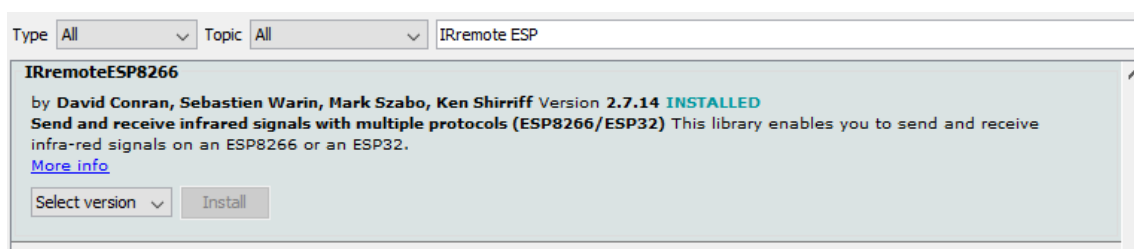
- Controle remoto de lâmpadas (liga/desliga ou até controle de brilho),
- Acionamento remoto de aparelhos,
- Controle remoto de alarmes,
- Controle remoto de câmera fotográfica.

A montagem do circuito é bem simples, interligando um sensor Receptor IR AX-1838Hs ao NodeMCU. A alimentação do sensor é fornecida pelo 3,3V do NodeMCU. E o pino de saída dos pulsos do Receptor é conectado ao pino D11 do NodeMCU. O Led verde está conectado no pino (porta), o Led azul no pino (porta) e o Led vermelho no pino (porta) do NodeMCU. Todos Leds tem um resistor de 220 Ω em série com as ligações. O lado chanfrado do Led é o cátodo, que deve ser conectado ao pino terra (GND).

Diagrama do Controle Remoto IR para NodeMCU:

Instalando a Biblioteca IRremote:

Para instalar a Biblioteca **IRremote**, clique em:

Sketch > Incluir Biblioteca > Gerenciar Bibliotecas

Após abrir a janela do Gerenciador de Biblioteca, refine a busca digitando **IRremote ESP**. Na biblioteca **IRremoteESP8266** (de David Conran), clique em **More Info** e depois em **Instalar.** Após alguns segundos, ela será automaticamente instalada. Lembre-se que o seu computador precisa estar conectado na internet, para poder baixar a biblioteca. Após a instalação da Biblioteca, é necessário que feche e abra novamente o programa Arduino IDE.

O sketch do exemplo é bem interessante, pois com o uso do controle remoto poderá ligar ou desligar os LEDs coloridos. Se quiser acionar um equipamento, poderá substituir os LEDs por um módulo de relé.

- Tecla 1 controla Led Verde / pino (porta),
- Tecla 2 controla Led Azul / pino (porta),
- Tecla 3 controla Led Vermelho / pino (porta).

Código Controle Remoto IR - LEDs coloridos:

```
 // Exemplo 7 – Controle Remoto IR -  Receptor Universal IR
// Apostila Eletrogate - Automação Residencial
// Gustavo Murta

#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>                                  // Biblioteca IRemote

int RECV_PIN = 14;                                     // NodeMCU pino D5 conectado no
Receptor IR
IRrecv irrecv(RECV_PIN);                               // criando a instância
decode_results results;                                // declarando os resultados
bool LED6 = false,
     LED7 = false,
     LED8 = false;                                     // estado dos LEDs
int atraso = 250;                                      // atraso após ligar LED

void setup(){
  pinMode(12, OUTPUT);                                  // LED vermelho no pino D6
  pinMode(13, OUTPUT);                                  // LED azul no pino D7
  pinMode(15, OUTPUT);                                  // LED verde no pino D8
  irrecv.enableIRIn();                                  // Inicializa a recepção de códigos
}

void loop(){
  results.value = 0;                                   // zera os registradores
  if (irrecv.decode(&results))                         // se algum código for recebido
  {
    irrecv.resume();                                   // reinicializa o receptor
  }
  if (results.value == 0xFFB04F)                       // pressione tecla 3 para controlar
LED vermelho (D6)
  {
    LED6 = !LED6;                                      // alterna o estado do LED D6
    digitalWrite(12, LED6);                            // acende ou apaga LED vermelho (D6)
    delay(atraso);                                     // atraso de 250 ms
  }
  if (results.value == 0xFF9867)                       // pressione tecla 2 para controlar
LED azul (D7)
  {
    LED7 = !LED7;                                      // alterna o estado do LED D7
    digitalWrite(13, LED7);                            // acende ou apaga LED azul (D7)
    delay(atraso);                                     // atraso de 250 ms
  }
  if (results.value == 0xFF6897)                       // pressione tecla 1 para controlar
LED verde (D05)
  {
    LED8 = !LED8;                                      // alterna o estado do LED D8
    digitalWrite(15, LED8);                            // acende ou apaga LED verde (D8)
    delay(atraso);                                     // atraso de 250 ms
  }
}
```

Referências:

- http://blog.eletrogate.com/guia-completo-do-controle-remoto-ir-receptor-ir-para-arduino/
- https://learn.adafruit.com/using-an-infrared-library/overview
- http://labdegaragem.com/profiles/blogs/6223006:BlogPost:315534
- https://hetpro-store.com/TUTORIALES/control-ir-con-arduino/
- http://www.righto.com/search/label/ir

Exemplo 8 – Relógio RTC – DS1307

Módulo RTC DS1307:

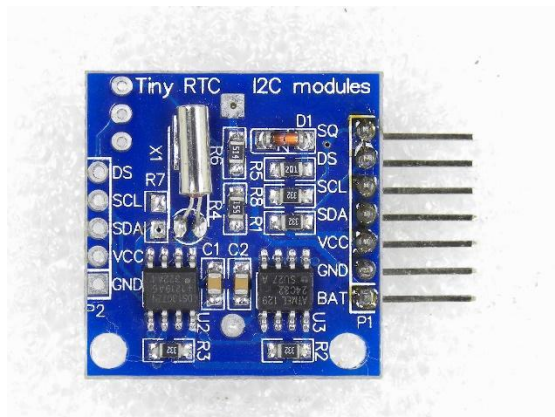
Esse módulo serve como relógio para projetos com o NodeMCU, Arduino e outros microcontroladores que precisam do controle do tempo. Pode ser usado como alarme despertador, controle automatizado de operações temporizadas, e claro como um relógio comum com horas e minutos. O módulo tem o chip DS1307 que é o relógio, uma EEPROM 24C32, um cristal de 32,768 KHz e uma pilha CR2032.

O chip DS1307 RTC (Real-Time Clock) é um relógio / calendário de baixa potência com codificação binária completa (BCD) mais 56 bytes de NV SRAM (memória estática RAM não-volátil). Os endereços e os dados são transferidos em série por meio do barramento bidirecional da interface I2C. O relógio / calendário fornece informações sobre segundos, minutos, horas, dia, data, mês e ano. O relógio funciona no formato de 24 ou 12 horas com o indicador AM / PM. O DS1307 possui um circuito de detecção de falha de energia que alterna automaticamente a alimentação para a pilha CR2032 (3V), na ausência de energia da fonte. A contagem de tempo do relógio é baseada no oscilador do cristal de 32,768 KHz.

DS1307 – folha de especificações (datasheet):
https://datasheets.maximintegrated.com/en/ds/DS1307.pdf

A memória EEPROM 24C32 permite o armazenamento de até 4096 KBytes e ela é não volátil, isto é, mesmo sem energia ela armazena os dados gravados. Para a comunicação com o microcontrolador ela também usa a interface I2C.

AT24C32 – folha de especificações (datasheet):
http://ww1.microchip.com/downloads/en/devicedoc/doc0336.pdf

Na placa do circuito do módulo, existe um espaço onde um sensor de temperatura DS18B20 poderá ser soldado. Esse sensor é opcional e não vem com o KIT.

Módulo RTC – DS1307

Pinagem do Módulo DS1307:

- SQ – Onda quadrada (normalmente não usado),
- DS – Sensor DS18B20 (opcional) ,
- SCL – Serial Clock (I2C Interface) ,
- SDA – Serial Data (I2C interface) ,
- VCC – Alimentação 5V (mínimo: 4,5V),
- GND – Terra,
- BAT – Tensão da bateria (aproximadamente 3V).

A alimentação do módulo RTC DS1307 deve ser de 5V. A tensão mínima de alimentação do chip DS1307 é de 4,5V. Portanto esse módulo não pode ser alimentado com 3,3V.

Os pinos de comunicação I2C do módulo já tem os resistores de pull up. Por isso não é necessário acrescentar esses resistores. Os pinos podem ser conectados diretamente no NodeMCU ESP8266.

A tensão da bateria (pilha de Lítio) deverá variar com o uso. Quando a pilha está nova, a tensão é de 3,2V aproximadamente.

Lista de Materiais:

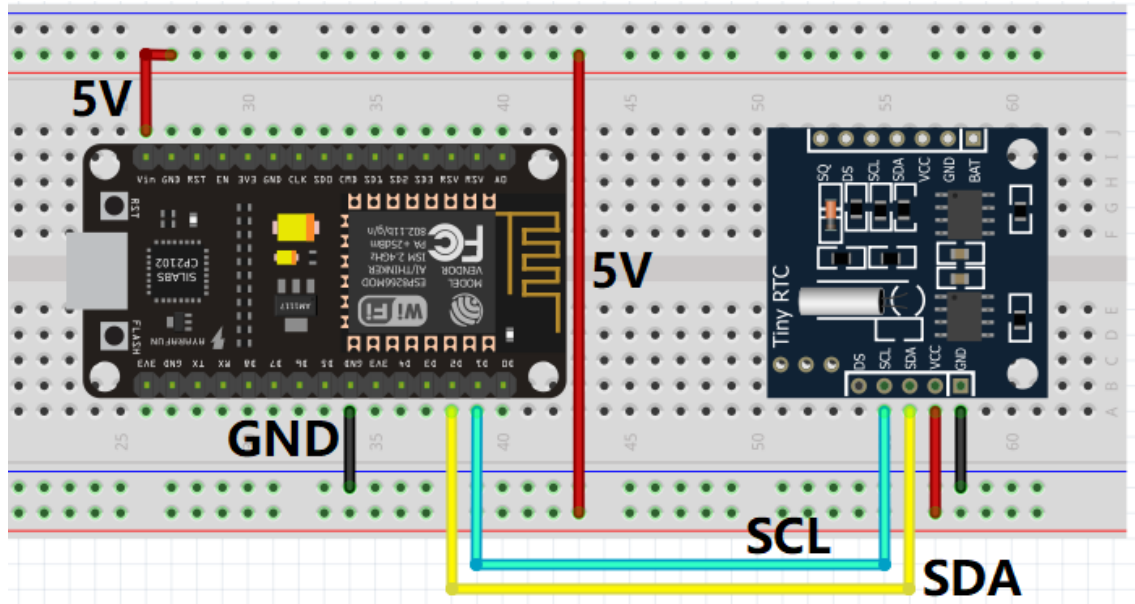
Para este exemplo você usará os seguintes componentes:

- NodeMCU ESP8266,
- Protoboard,
- Jumpers de ligação,
- Módulo RTC DS1307.

Montagem do Módulo RTC DS1307 com NodeMCU ESP8266:

Faça as ligações de acordo:

SCL RTC => SCL D1 VCC RTC => 5V NodeMCU

SDA RTC => SDA D2 GND RTC => GND NodeMCU

Instalando a Biblioteca RTClib:

A Biblioteca RTClib será usada nesse exemplo. Para instalar a nova biblioteca na IDE Arduino, clique em:

Sketch > Incluir Biblioteca > Gerenciar Bibliotecas

Após abrir a janela do Gerenciador de Biblioteca, refine a busca digitando RTClib. Na biblioteca RTClib, clique em More Info e depois em Instalar. Após alguns segundos, ela será automaticamente instalada. Lembre-se que o seu computador precisa estar conectado na internet, para poder baixar a biblioteca. Após a instalação da Biblioteca, é necessário que feche e abra novamente o programa Arduino IDE.

Esse é o link da Biblioteca RTClib:

https://github.com/adafruit/RTClib

Código Arduino – Módulo RTC DS1307:

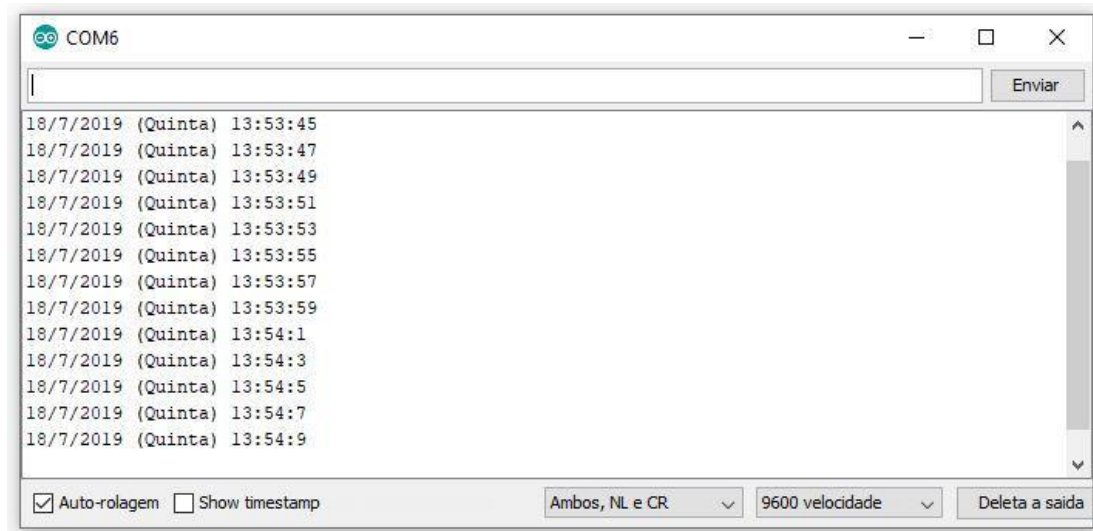
```
// Exemplo 8 – Módulo RTC DS1307 com NodeMCU ESP8266
// Apostila Eletrogate - KIT Automação Residencial
// Gustavo Murta
// https://github.com/adafruit/RTClib/blob/master/examples/ds1307/ds1307.ino

#include "RTClib.h"                         // biblioteca RTClib

RTC_DS1307 rtc;                             // criando a instância RTC

char diasDaSemana[7][12] = {"Domingo", "Segunda", "Terca", "Quarta", "Quinta", "Sexta",
"Sabado"};

void setup (){
  Serial.begin(115200);                                // monitor da console serial
  if (! rtc.begin())                                   // se o RTC não foi inicializado
  {
    Serial.println("RTC nao pode ser encontrado!");    // imprime mensagem
    while (1);
  }

  if (! rtc.isrunning())                               // se o RTC não estiver rodando
  {
    Serial.println("RTC nao esta funcionando!");       // imprime mensagem
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));    // ajusta relógio com o relógio do
seu PC
  }
}

void loop (){
  DateTime now = rtc.now();                           // faz a leitura dos dados do RTC
  Serial.print(now.day(), DEC);                       // imprime o dia do mês
  Serial.print('/');
  Serial.print(now.month(), DEC);                     // imprime o mês
  Serial.print('/');
  Serial.print(now.year(), DEC);                      // imprime o ano
  Serial.print(" (");
  Serial.print(diasDaSemana[now.dayOfTheWeek()]);     // imprime o dia da semana
  Serial.print(") ");
  Serial.print(now.hour(), DEC);                      // imprime horas
  Serial.print(':');
  Serial.print(now.minute(), DEC);                    // imprime minutos
  Serial.print(':');
  Serial.print(now.second(), DEC);                    // imprime segundos
  Serial.println();                                   // imprime uma linha
  delay(2000);                                        // atraso de 2 segundos
}
```

Esse programa foi traduzido e adaptado do exemplo DS1307 da biblioteca RTClib. Ele acerta a data e as horas com o relógio do seu PC. E fica imprimindo a data e as horas, a cada 2 segundos. Esse exemplo é para você entender basicamente como o relógio RTC pode ser usado.

Sabendo-se que o módulo tem uma memória EEPROM e pode ainda ter um sensor de temperatura (opcional), você poderá criar inúmeras aplicações bem interessantes.

É recomendado que estude as referências abaixo para poder desenvolver novos projetos e fazer novas montagens.

Veja a data e o horário na console serial (115200 Bps):

Referências:

- https://blog.eletrogate.com/rtc-real-time-clock-ds1302-1307-e-3231/
- https://learn.adafruit.com/ds1307-real-time-clock-breakout-board-kit/overview
- https://wiki.dfrobot.com/Real_Time_Clock_Module__DS1307___SKU_DFR0151_
- https://howtomechatronics.com/projects/arduino-touch-screen-music-player-alarm-clock-project/

Exemplo 9 - Módulo RFID MFRC522

Os cartões RFID são usados em uma série de aplicações na vida cotidiana, desde cartões de crédito até cartões de identificação automática para controle de acesso.

Esses cartões possuem uma tarja magnética que armazena dados digitais e podem ser lidos e escritos por meio de ondas de rádio.

RFID é a sigla para: Radio Frequency Identification. É uma tecnologia que pertence a uma área chamada de Automatic Identification and Data Capture (AIDC).

Os métodos AIDC identificam objetos e pessoas automaticamente, coletam dados sobre eles e entram com esses dados diretamente em softwares com pouca ou nenhuma intervenção humana.

Módulo RFID MFRC522

O módulo leitor RFID baseado no chip MFRC522 é utilizado para comunicação sem fio a uma frequência de 13,56 MHz. A empresa que desenvolveu o módulo é a NXP, uma das gigantes do setor de semicondutores.

Trata-se de um chip de consumo extremamente baixo e dimensões reduzidas. A sua principal aplicação é a leitura de cartões que utilizam o padrão Mifare.

As tags RFID podem ser usadas para ler várias informações sobre o portador do cartão, permitindo sua identificação automática para validação ou não do acesso.

Especificações do MFRC522

- Corrente de trabalho: 13-26mA / DC 3.3V;
- Pico de corrente: <30mA;
- Frequência de operação: 13,56MHz;
- Tipos de cartões suportados: Mifare1 S50, S70 Mifare1, Mifare UltraLight, Mifare Pro, Mifare Desfire;
- Parâmetro de Interface SPI;
- Taxa de transferência: 10 Mbit/s.

Como funciona o módulo RFID?

Métodos e tecnologias que usam RFID baseiam-se em ondas de rádio para ler e gravar dados. Os sistemas RFID são feitos dos seguintes componentes principais:

- RFID tag + Antena: Um pequeno circuito integrado embutido no cartão e um antena são usados para armazenar e transmitir dados para o leitor RFID. A Identificação da tag é chamada de UID, trata-se de um número hexadecimal característico de cada cartão ou chaveira RFID;
- Leitor RFID: O leitor converte as ondas de rádio transmitidas pela antena da tag para um formato que possa ser usado internamente no computador;

O módulo leitor também faz parte do kit e é por meio dele que é feita a gravação/leitura de dados. No nosso caso, o leitor é baseado no CI MFRC522.

Ao se aproximar o cartão do módulo leitor, a antena transmite os dados na frequência de operação (13,56MHz) para o leitor, que recebe os dados e converte para um formato que possa ser processado pelo CI e manipulado por um MCU externo (Arduino, Raspberry ou outro sistema qualquer).

Descrição do Projeto: RFID com NodeMCU

Confira o passo a passo da descrição do nosso projeto de sistema de controle de acesso utilizando um Kit RFID com Arduino:

- Montar um circuito para ler um cartão RFID.
- Fazer um software com Arduino para identificar a tag do cartão e apresentar uma mensagem num display LCD de “Acesso Liberado” para a tag salva, ou “Acesso negado”, para qualquer outra tag.
- Um buzzer também será inserido no circuito para avisar e emitir beeps para cada situação.

Aspectos de Hardware

Vamos precisar dos seguintes materiais para montar o circuito:

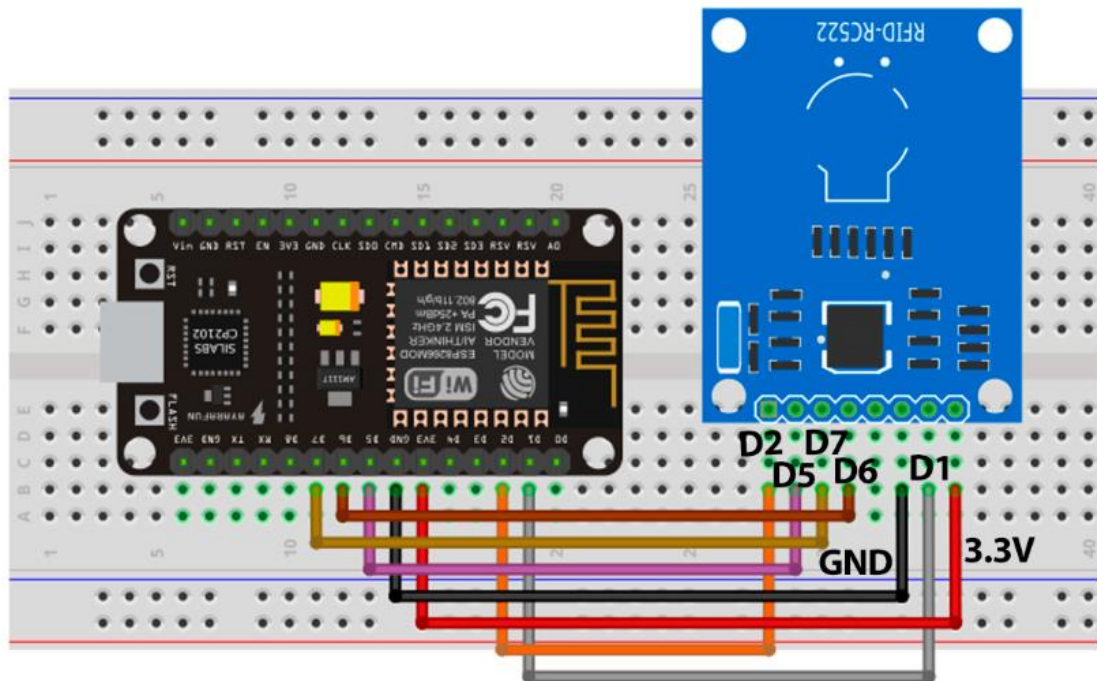
- NodeMCU ESP8266;
- KIT RFID;
- Display LCD 16×2;
- Potenciômetro de 10KΩ;
- Buzzer ativo;
- Protoboard;
- Jumpers para protoboard;

De posse de todos os componentes, você pode montar o circuito conforme o diagrama abaixo. A pinagem do RC522 com NodeMCU é a seguinte:

- SDA -> 9
- SCK -> 13
- MOSI -> 11
- MISO -> 12
- RST -> 8

<u>Lembre-se de ligar o Vcc do RFID-RC522 no 3.3V!</u>

Daí a necessidade de usar o level shifter bidirecional entre um e outro. Alguns canais são unidirecionais, mas como você já vai usar um circuito de level shifter, recomendamos já comprar um totalmente bidirecional, como o indicado na lista de componentes.

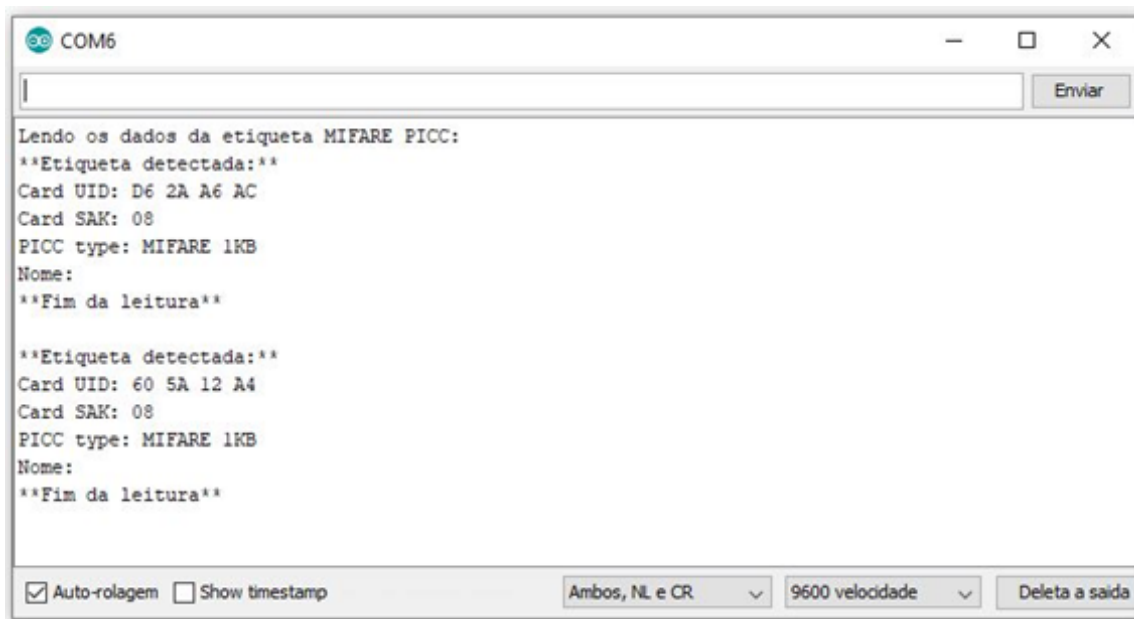
Diagrama do Módulo RFID MFRC522 com NodeMCU

Código do Projeto:

```
 // Exemplo 9 – Modulo RFID com Arduino

// Apostila Eletrogate - Kit Automação Residencial


#include <SPI.h>                                    // bilblioteca SPI
#include <MFRC522.h>                                // biblioteca MFRC522
#define RST_PIN 5                                   // pino Reset = Arduino D9
#define SS_PIN  4                                  // pino SS(SPI) = Arduino D10
MFRC522 mfrc522(SS_PIN, RST_PIN);                   // cria a instancia MFRC522

void setup()
{
  Serial.begin(115200);                                              // console serial 9600 bps
  SPI.begin();                                                     // inicializa interface SPI
  mfrc522.PCD_Init();                                              // inicializa o modulo
MFRC522
  Serial.println(F("Lendo os dados da etiqueta MIFARE PICC:"));    // imprime mensagem console
serial
}

void loop()
{
  MFRC522::MIFARE_Key key;                                 // preparando as chaves
  for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;      // montando as chaves

  byte block;                                              // algumas variaveis necessarias
  byte len;
  MFRC522::StatusCode status;                              // lendo o estado das etiquetas
  if ( ! mfrc522.PICC_IsNewCardPresent())                  // se nao houver etiqueta
  {
    return;
  }
  if ( ! mfrc522.PICC_ReadCardSerial())                    // selecione uma das etiquetas
  {
    return;
  }
// Continua na próxima página
```

```
  Serial.println(F("**Etiqueta detectada:**"));          // imprime mensagem
  mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));       // mostra informacoes da etiqueta
  Serial.print(F("Nome: "));                              // imprime mensagem

  byte buffer1[18];
  block = 4;
  len = 18;
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 4, &key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK)                        // se falhar na leitura
  {
    Serial.print(F("Falha na autenticacao: "));            // imprime mensagem
    Serial.println(mfrc522.GetStatusCodeName(status));     // imprime o erro
    return;
  }
  status = mfrc522.MIFARE_Read(block, buffer1, &len);
  if (status != MFRC522::STATUS_OK)                        // se falhar na leitura
  {
    Serial.print(F("Falha na leitura: "));                 // imprime mensagem
    Serial.println(mfrc522.GetStatusCodeName(status));     // imprime o estado
    return;
  }
  for (uint8_t i = 0; i < 16; i++)                         // para 15 ponteiros
  {
    if (buffer1[i] != 32)                                  // se o buffer for diferente de 32
    {
      Serial.write(buffer1[i]);                            // imprime o buffer
    }
  }
  Serial.print(" ");
  Serial.println(F("\n**Fim da leitura**\n"));             // imprime mensagem
  delay(1000);                                             // atraso de 1 segundo
  mfrc522.PICC_HaltA();                                    // para a leitura do RFID
  mfrc522.PCD_StopCrypto1();
}
```

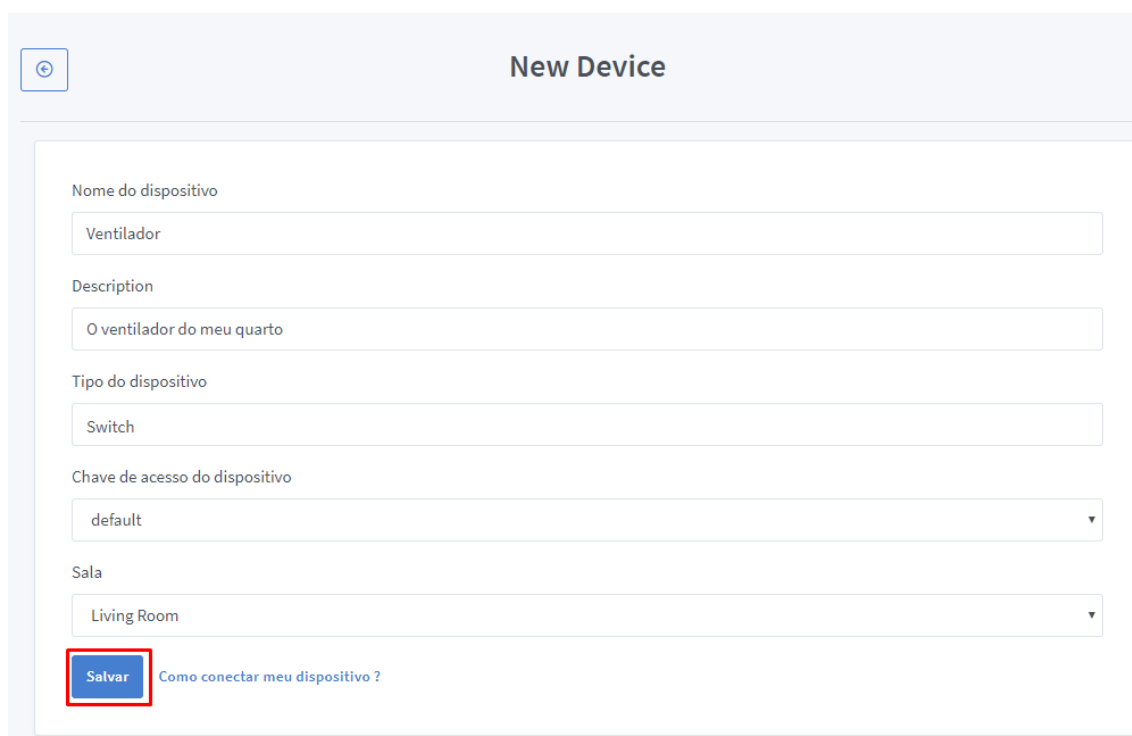
Janela da console serial da IDE Arduino – um cartão e um chaveiro foram lidos.

Referências:

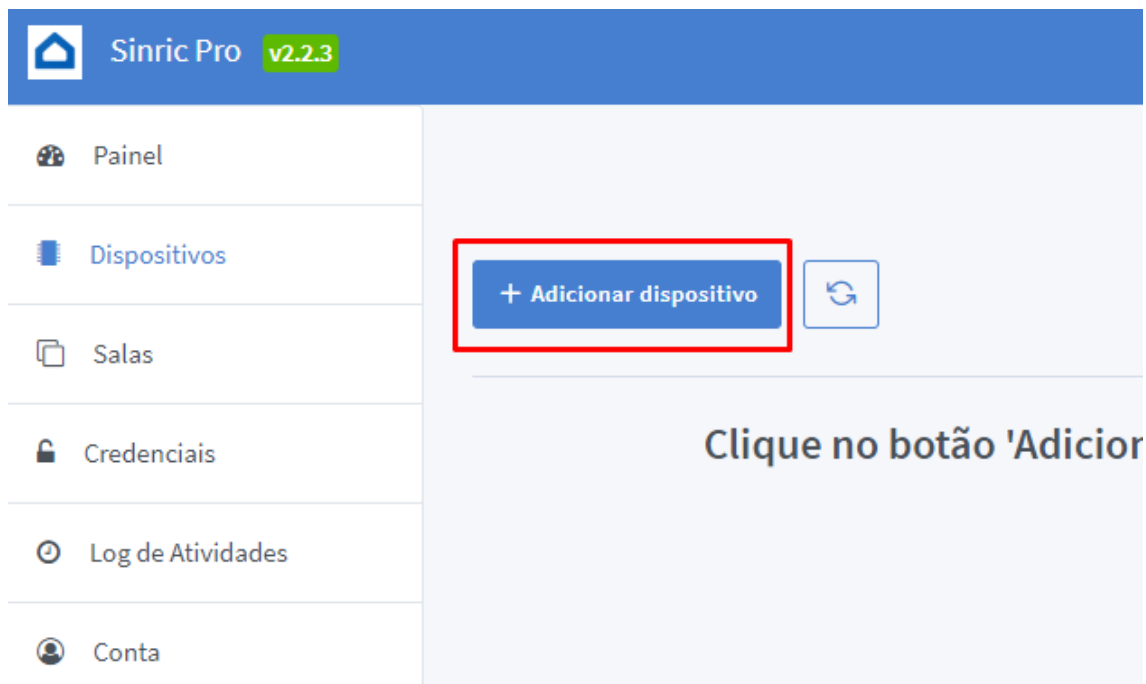
- https://blog.eletrogate.com/guia-basico-da-nfc-para-arduino/
- https://blog.eletrogate.com/kit-rfid-com-arduino-sistema-de-controle-de-acesso/
- https://randomnerdtutorials.com/security-access-using-mfrc522-rfid-reader-with-arduino/
- https://howtomechatronics.com/tutorials/arduino/rfid-works-make-arduino-based-rfid-door-lock/

Exemplo 10 - Conectando o ESP8266 a Amazon Alexa

O conceito da automação residencial, ou domótica, surgiu na França na década de 80 do século passado, e até aqui percorreu um longo caminho para se tornar acessível e viável. Neste post de hoje você vai aprender bem detalhadamente como automatizar sua casa de uma forma incrível, barata e o melhor: você não precisará de comprar nenhum "equipamento smart" (como lâmpadas ou tomadas), que são pouco acessíveis no mercado brasileiro, pois tudo será feito por você mesmo!

O primeiro passo para a conexão é fazer o download do aplicativo Alexa em sua loja de aplicativos preferida e criar (ou logar) sua conta.

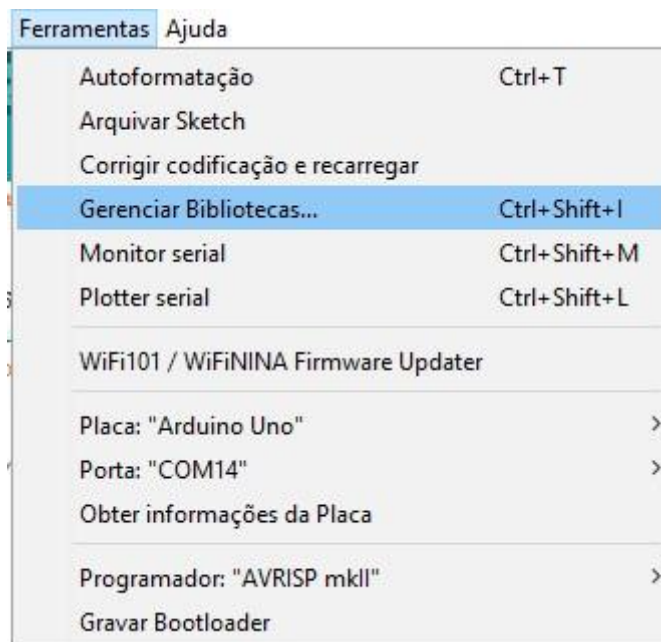
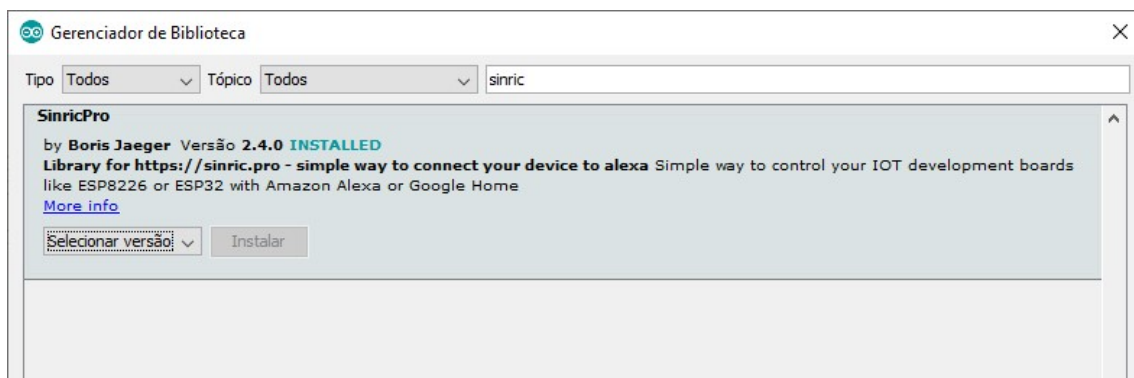
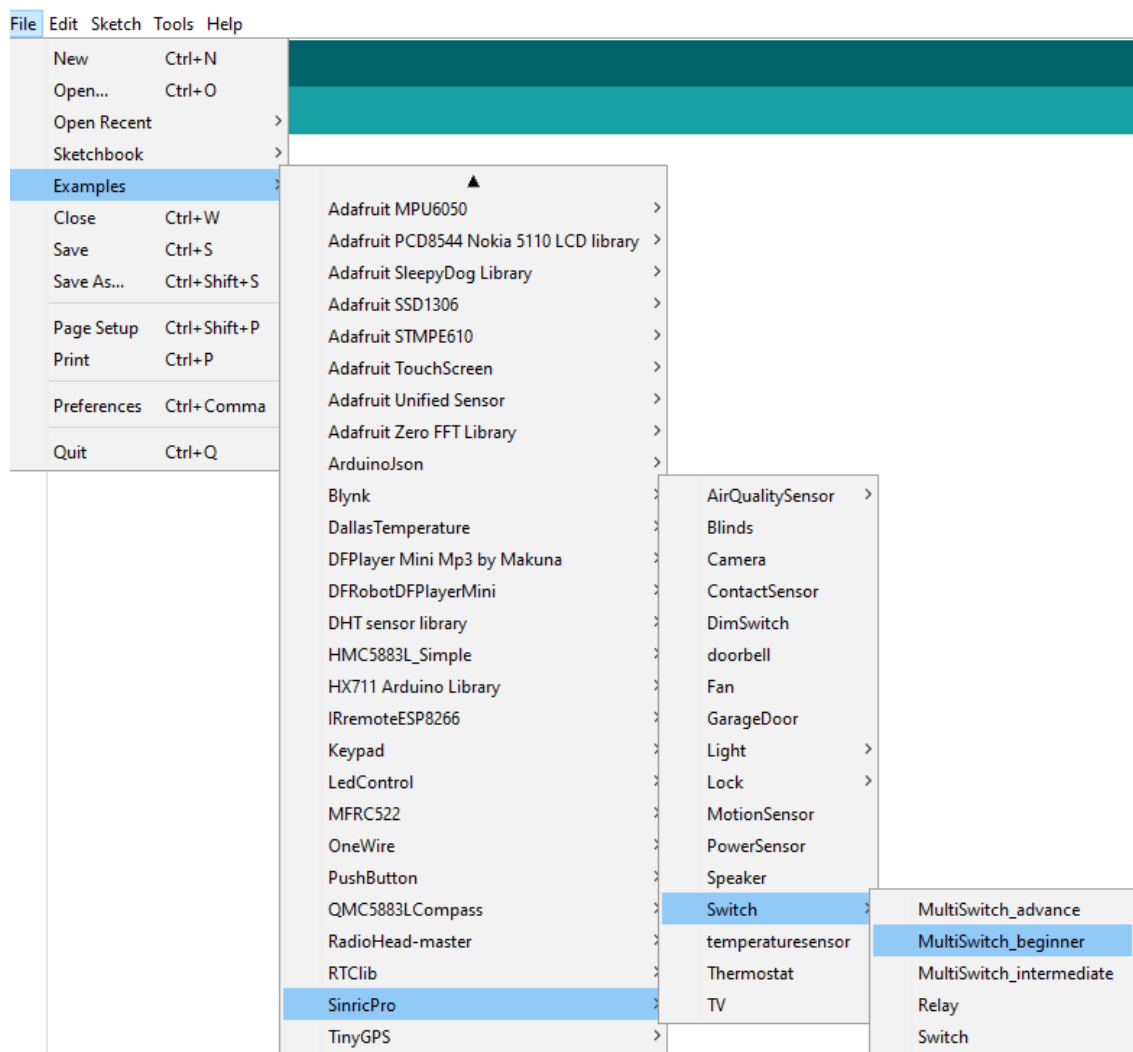
Após realizar as configurações do aplicativo, vá até o site da Sinric Pro e crie uma conta:

https://sinric.pro/pt-index.html

Essa conta te dará direito a ter dispositivos grátis, que poderão variar segundo a política da empresa.

Após a criação da conta no site da Sinric, vá em “Dispositivos”, na lateral esquerda, e clique em adicionar dispositivo.

Selecione um nome para o dispositivo, descrição e o tipo de dispositivo selecione: **Switch.** Salve o dispositivo criado.

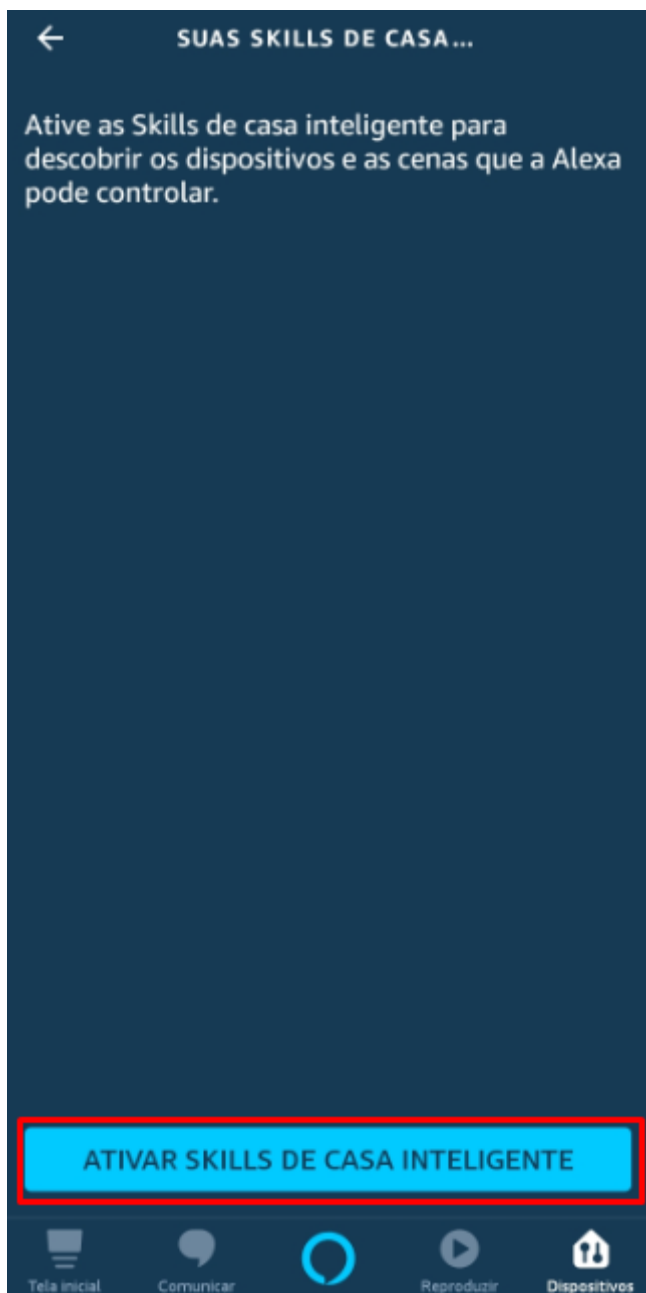
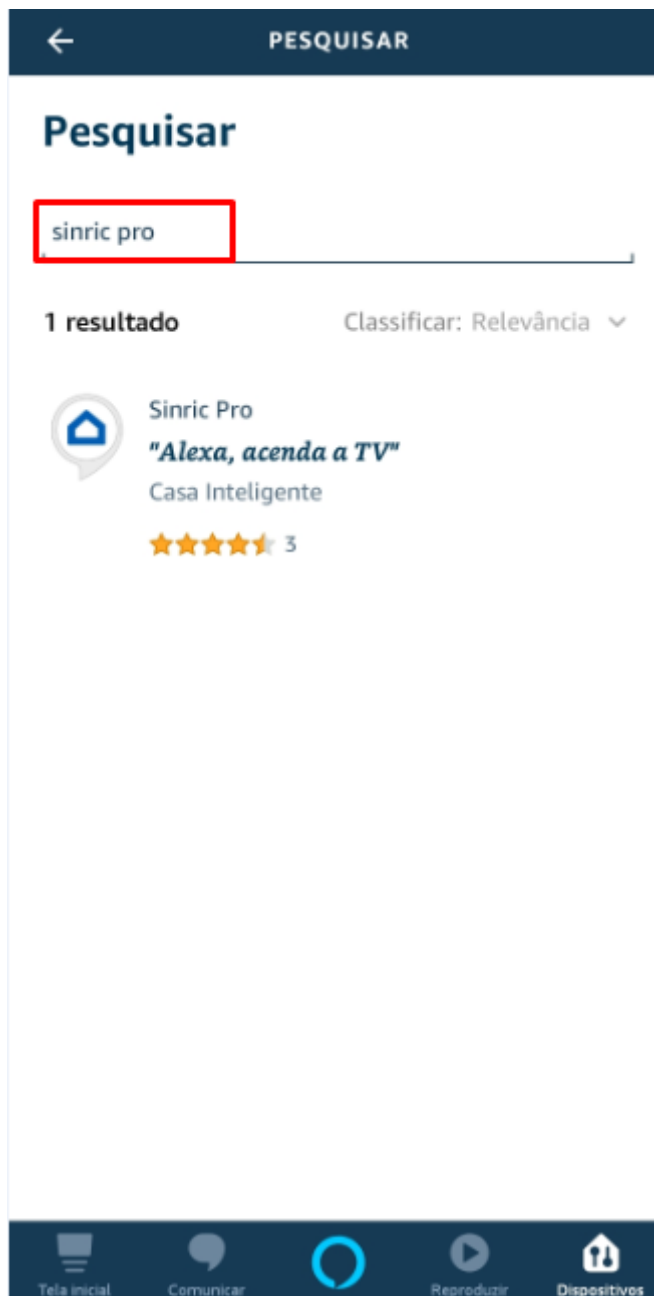
O dispositivo possui um ID, e este ID será usado no código da Arduino IDE. Crie os dispositivos que você deseja.

Na lateral esquerda vá em “Credenciais” e copie a chave do App e a senha do App.

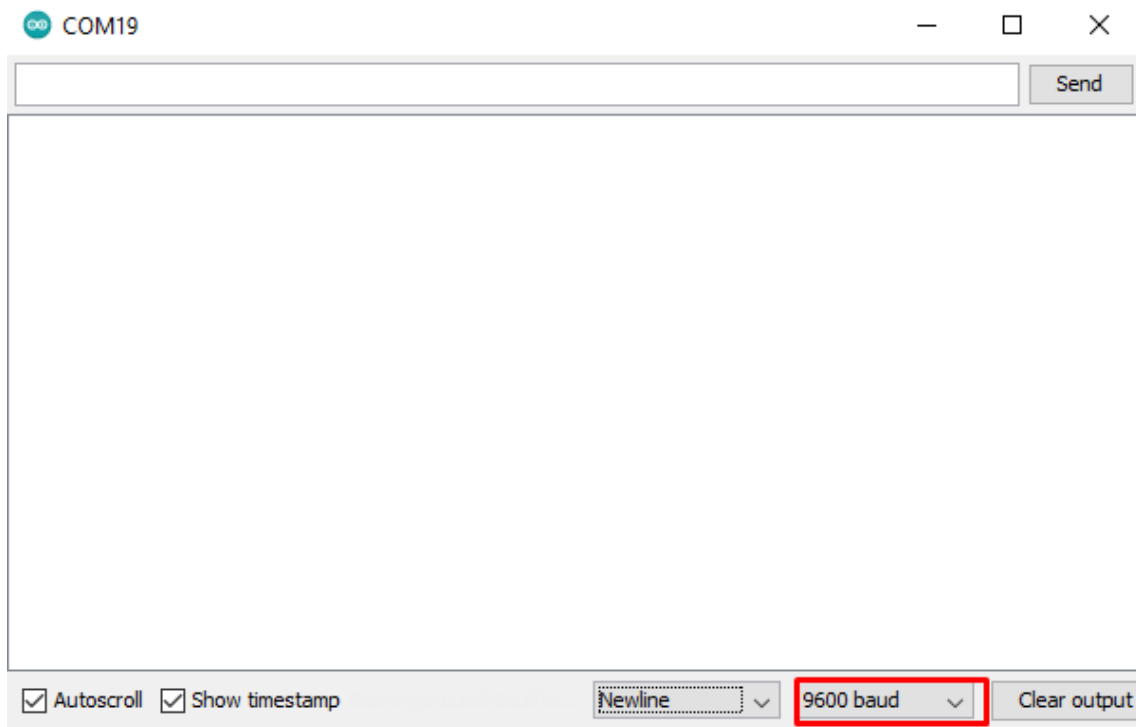
Na Arduino IDE, vá em Ferramentas > Gerenciar Bibliotecas

Instale a biblioteca SinricPro, do Boris Jaeger.

Após isso vá em “Arquivo > Exemplos > SinricPro > Switch > MultiSwitch_beginner”

Substitua os dados abaixo no código com o nome do seu WiFi, senha do WiFi, App_Key e App_Secret (dados encontrados na aba de credenciais do Sinric Pro), e insira os IDs dos dispositivos.

```
#define WIFI_SSID         "YOUR-WIFI-SSID"
#define WIFI_PASS         "YOUR-WIFI-PASSWORD"
#define APP_KEY           "YOUR-APP-KEY"      // Should look like "de0bxxxx-1x3x-4x3x-ax2x-5dabxxxxxxxx"
#define APP_SECRET        "YOUR-APP-SECRET"   // Should look like "5f36xxxx-x3x7-4x3x-xexe-e86724a9xxxx-4c4axxxx-3x3x-x5xe-x9x3-333d65xxxxxx"

#define SWITCH_ID_1       "YOUR-DEVICE-ID"    // Should look like "5dc1564130xxxxxxxxxxxxxx"
#define SWITCH_ID_2       "YOUR-DEVICE-ID"    // Should look like "5dc1564130xxxxxxxxxxxxxx"
#define SWITCH_ID_3       "YOUR-DEVICE-ID"    // Should look like "5dc1564130xxxxxxxxxxxxxx"
#define SWITCH_ID_4       "YOUR-DEVICE-ID"    // Should look like "5dc1564130xxxxxxxxxxxxxx"
```

Após isso, vá no aplicativo Alexa, vá Dispositivos > Suas Skills de Casa Inteligente.

Vá em “Ativar Skills de Casa Inteligente” e procure por “Sinric Pro”

Após isso ative a Skill e faça login com a conta Sinric criada.

Após isso vá em "Descobrir Dispositivos", no aplicativo Alexa, e seus dispositivos do ESP será encontrado.

Agora, via Alexa, você poderá acionar por voz seus dispositivos. Para visualizar a comunicação com o NodeMCU, abra o Monitor Serial, **coloque o Baud Rate em 9600** e você verá o seu dispositivo recebendo a informação da Nuvem.

Para incluir o acionamento das portas do seu NodeMCU, basta incluir o que deve ser feito neste trecho de código:

```
bool onPowerState1(const String &deviceId, bool &state) {
  Serial.printf("Device 1 turned %s\r\n", state?"on":"off");
  return true; // request handled properly

}

bool onPowerState2(const String &deviceId, bool &state) {
  Serial.printf("Device 2 turned %s\r\n", state?"on":"off");
  return true; // request handled properly

}

bool onPowerState3(const String &deviceId, bool &state) {
  Serial.printf("Device 3 turned %s\r\n", state?"on":"off");
  return true; // request handled properly

}

bool onPowerState4(const String &deviceId, bool &state) {
  Serial.printf("Device 4 turned %s\r\n", state?"on":"off");
  return true; // request handled properly

}
```

Neste trecho você poderá inserir os comandos “digitalWrite”, como aprendido ao longo da apostila.

Para mais exemplos acesse:

- https://blog.eletrogate.com/automacao-residencial-com-alexa-amazon-e-nodemcu/
- https://blog.eletrogate.com/criando-dispositivos-inteligentes-para-alexa-amazon-1/
- https://blog.eletrogate.com/controlando-o-brilho-da-lampada-com-a-alexa/

Considerações finais

Essa apostila tem por objetivo apresentar alguns exemplos básicos sobre como utilizar os componentes do Kit Automação Residencial, a partir dos quais você pode combinar e fazer projetos mais elaborados por sua própria conta.

Nas seções de referências de cada exemplo e nas referências finais, também tentamos indicar boas fontes de conteúdo objetivo e com projetos interessantes. Sobre esse ponto, que consideramos fundamental, gostaríamos de destacar algumas fontes de conhecimento que se destacam por sua qualidade.
O fórum oficial Arduino possui muitas discussões e exemplos muito bons. A comunidade de desenvolvedores é bastante ativa e certamente pode te ajudar em seus projetos. No Project Hub poderá encontrar milhares de projetos com Arduino.

- Fórum oficial Arduino*:* *https://forum.arduino.cc/*
- Project Hub Arduino : *https://create.arduino.cc/projecthub*

O Instructables também é uma ótima referência do mundo maker atual. Pessoas que buscam construir suas próprias coisas e projetos encontram referências e compartilham suas experiências no site.

- Instructables: *https://www.instructables.com/*

O Hackster.IO e o Hackaday.IO também têm inúmeros projetos bem interessantes.

- Hackster.IO: *https://www.hackster.io/projects*
- Hackaday.IO: *https://hackaday.io/projects*

O Maker pro é outro site referência no mundo em relação aos projetos com Arduino. Há uma infinidade de projetos, todos bem explicados e com bom conteúdo.

- Maker pro*:* *https://maker.pro/projects/arduino*

Em relação à eletrônica, teoria de circuitos e componentes eletrônicos em geral, deixamos alguns livros essenciais na seção final de referências. O leitor que quer se aprofundar no mundo da eletrônica certamente precisará de um livro basilar e de bons conhecimentos em circuitos eletro-eletrônicos.

No mais, esperamos que essa apostila seja apenas o início de vários outros projetos e, quem sabe, a adoção de Kits mais avançados, como o de **Robótica**. Qualquer dúvida, sugestão, correção ou crítica a esse material, fique à vontade para relatar em nosso blog oficial:

http://blog.eletrogate.com/

Referências gerais

1. Fundamentos de Circuitos Elétricos. Charles K. Alexander; Matthew N. O. Sadiku. Editora McGraw-Hill.
2. Circuitos elétricos. James W. Nilsson, Susan A. Riedel. Editora: Pearson; Edição: 8.
3. Microeletrônica - 5ª Ed. - Volume Único (Cód: 1970232). Sedra,Adel S. Editora Pearson.
4. Fundamentals of Microelectronics. Behzad Razavi. Editora John Wiley & Sons; Edição: 2nd Edition (24 de dezembro de 2012).

Abraços e até a próxima!

Vitor Vidal
Engenheiro eletricista, mestrando em eng. elétrica e apaixonado por eletrônica, literatura, tecnologia e ciência. Divide o tempo entre pesquisas na área de sistemas de controle, desenvolvimento de projetos eletrônicos e sua estante de livros.

Partes editadas por Vitor Vidal :
Parte I - Revisão de circuitos elétricos e componentes básicos
Parte III - Seção de Exemplos Práticos - Exemplos 1 ao 8 e 10.

Gustavo Murta
Consultor e Projetista de Sistemas Embarcados.
Técnico em eletrônica, formado em Curso superior de TPD, pós-graduado em Marketing. Trabalhou por muitos anos na IBM na área de manutenção de computadores de grande porte. Aposentou-se, podendo curtir o que mais gosta : estudar e ensinar Tecnologia. Hobbista em eletrônica desde 1976. Gosta muito de Fotografia e Observação de aves.

Revisão de todas as partes editadas por Vitor Vidal.
Partes editadas por Gustavo Murta :

Exemplo 6 - Display LCD 16x2
Exemplo 7 – Controle Remoto IR + Receptor Universal IR
Exemplo 8 – Relógio RTC – DS1307

Gustavo Nery
Cursando Engenharia de Controle e Automação pela UFMG. Apaixonado por eletrônica, computação e tecnologias na área de sistemas embarcados. Nos tempos livres me divido entre desenvolver pesquisa na universidade, adquirir novos conhecimentos e estar com a família.

Revisão de todas as partes.
Partes editadas por Gustavo Nery:
Exemplo 10 - Conectando o ESP8266 a Amazon Alexa

Ricardo Lousada
Graduando em Engenharia de Controle e Automação pela UFMG. Ocupo meu tempo aprendendo cada vez mais sobre eletrônica e programação, áreas que mais gosto. Meus hobbies são cinema e livros.

Revisão de todas as partes.
Partes editadas por Ricardo Lousada:
Parte II - NodeMCU ESP8266
Diagramas de todos os exemplos
Exemplo 5 - Módulo Relé 4 canais
Exemplo 9 - Módulo RFID MFRC522.

Versão 1.0 – Janeiro de 2021

APOSTILA KIT

AUTOMAÇÃO RESIDENCIAL

Esta apostila acompanha o **Kit AUTOMAÇÃO RESIDENCIAL** da Eletrogate, e contém conteúdos relacionados a todos os componentes do Kit.

WWW.ELETROGATE.COM